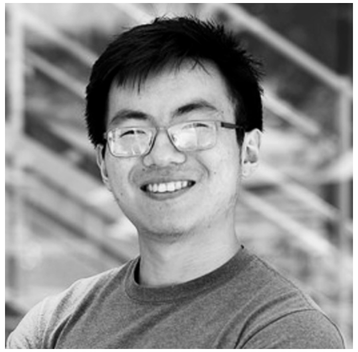
MacroBase: A Search Engine for Fast Data

Firas Abuzaid

Peter Bailis, Jialin Ding, Edward Gan, Kexin Rong, Sahaana Suri

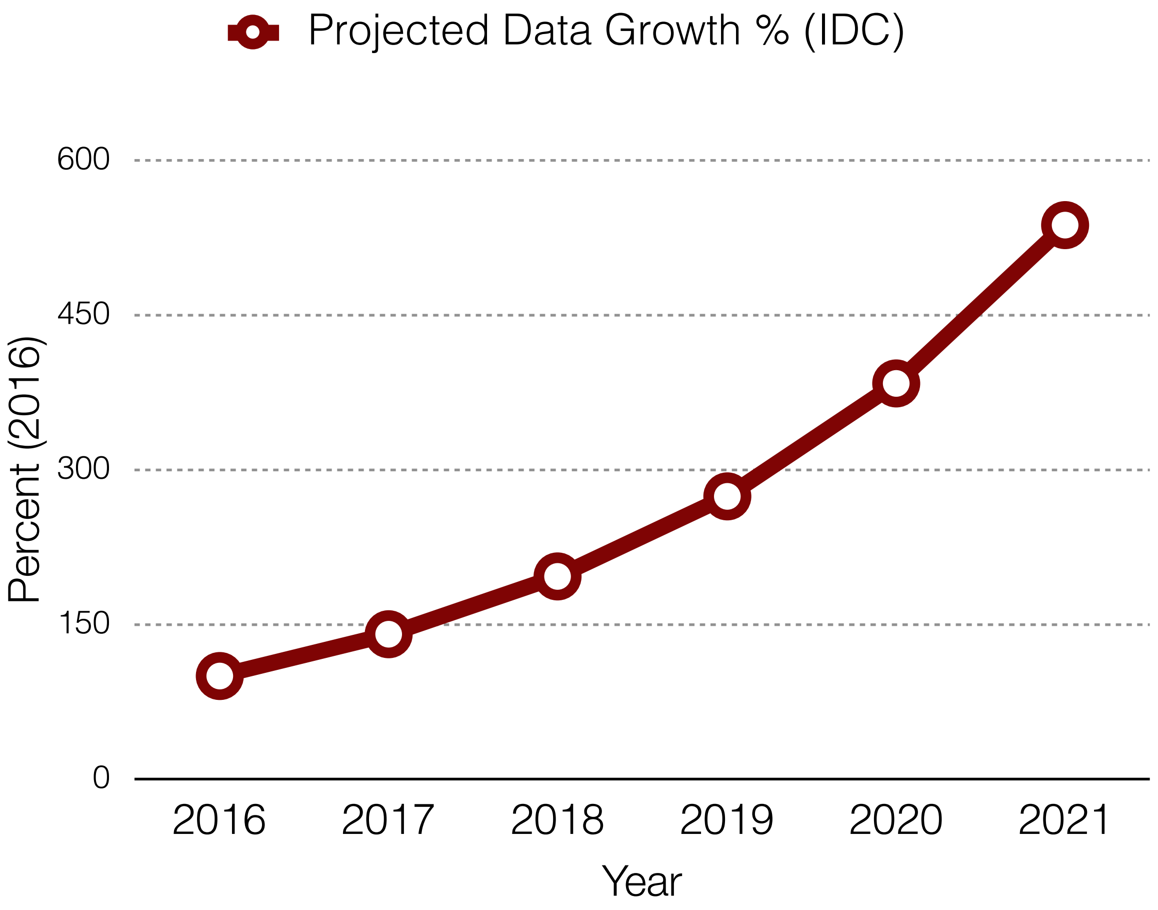
STANFORD INFOLAB

Team MacroBase

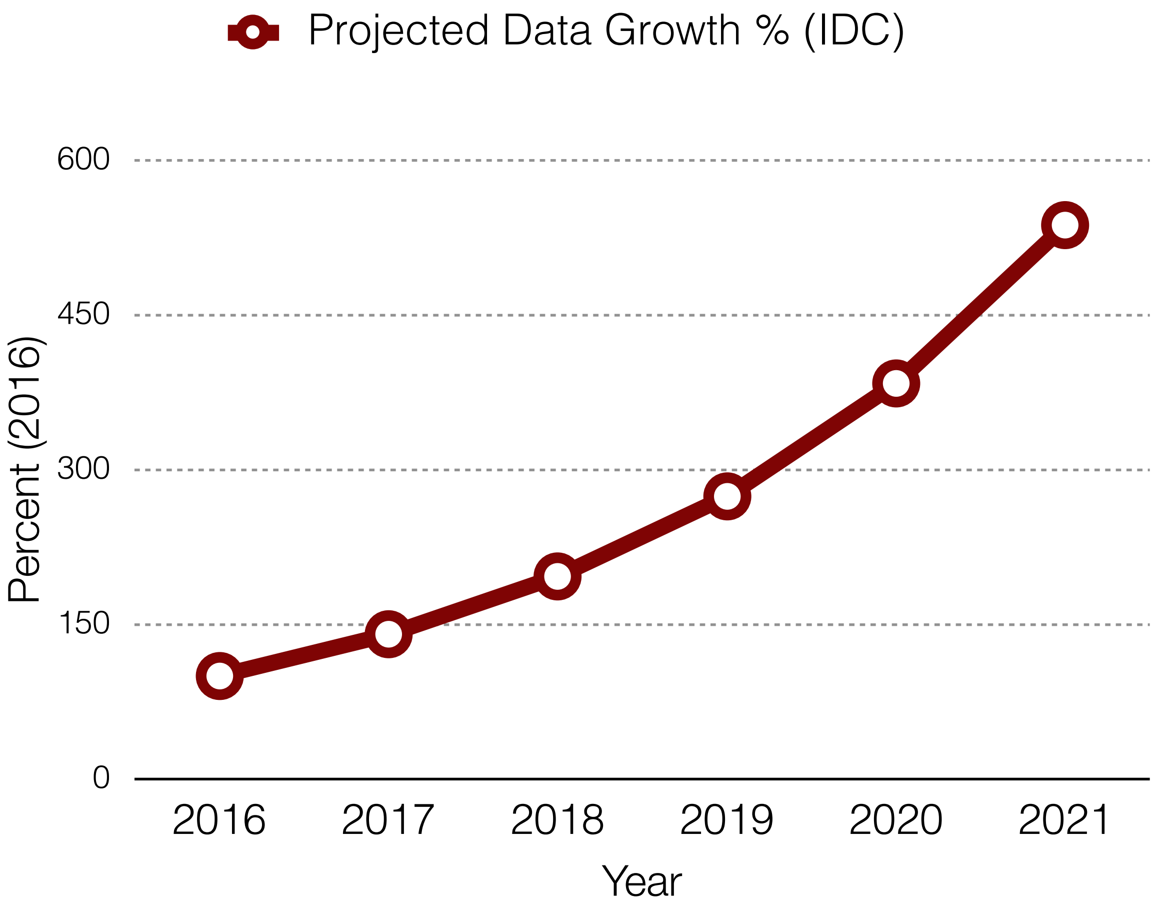
					
Peter Bailis Professor	Edward Gan 3rd year Ph.D.	Kexin Rong 3rd year Ph.D.	Sahaana Suri 3rd year Ph.D.	Firas Abuzaid 3rd year Ph.D.	Jialin Ding 4th year B.S.

macrobase@cs.stanford.edu

Monitoring & Telemetry Drive Data Volumes

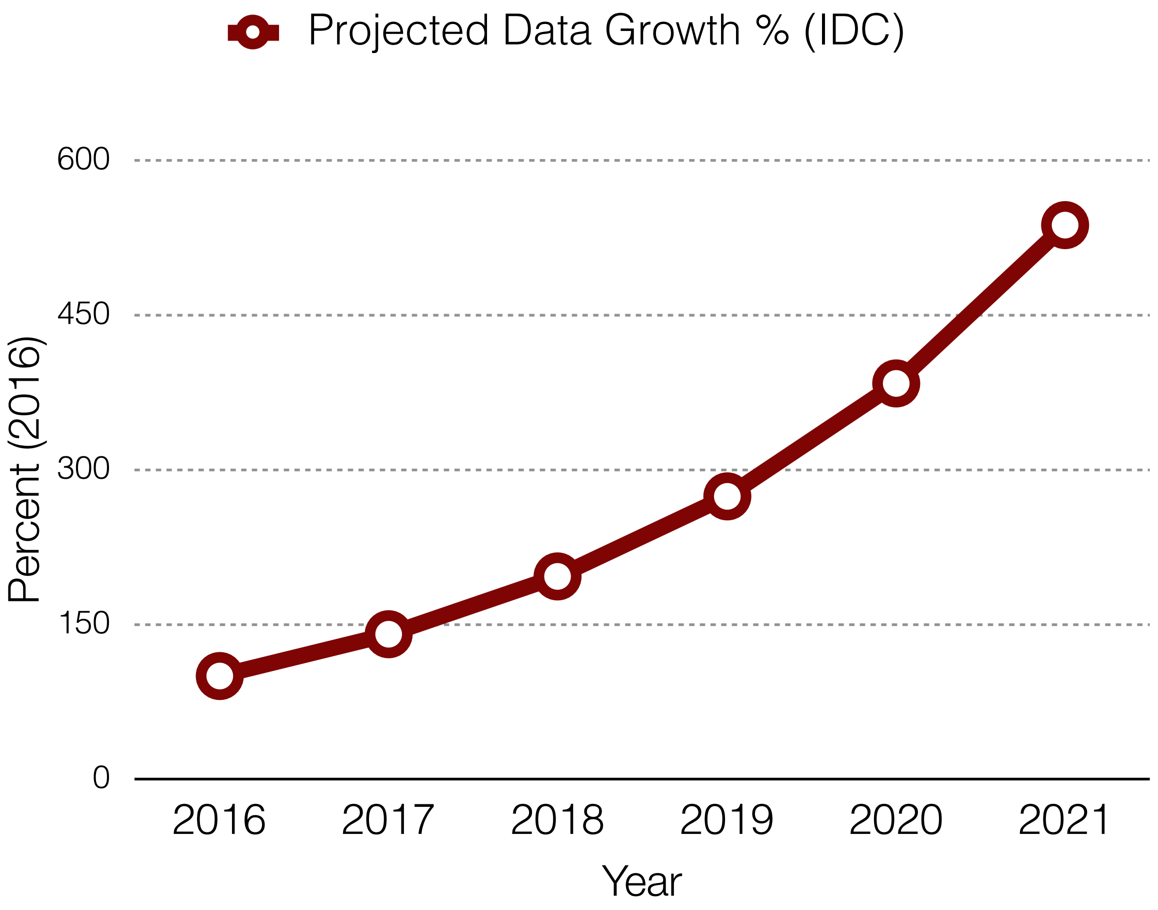


Monitoring & Telemetry Drive Data Volumes



Ability + need to monitor complex applications relying on sensors, processes, production telemetry

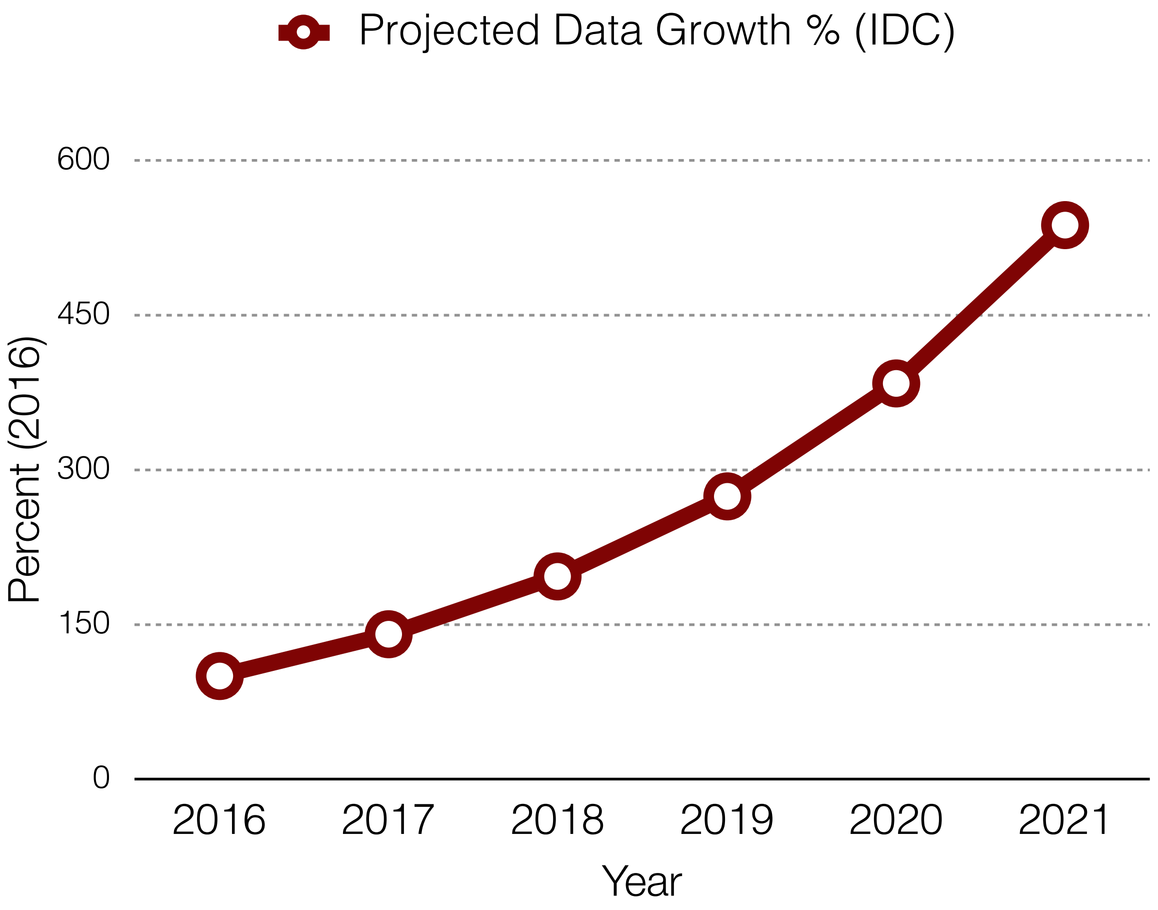
Monitoring & Telemetry Drive Data Volumes



Ability + need to monitor complex applications relying on sensors, processes, production telemetry

Reduced storage costs due to Big Data systems (e.g., HDFS, S3, Kafka), cloud

Monitoring & Telemetry Drive Data Volumes

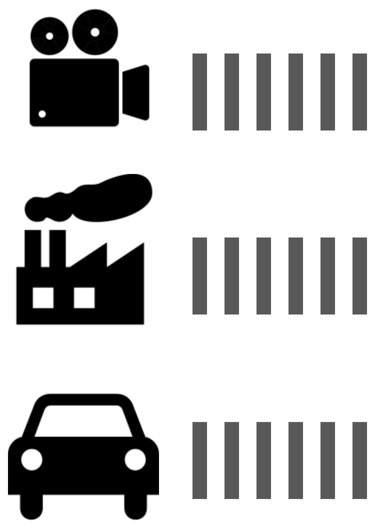


Data volumes continue to grow; storage and compute cheaper and easier than ever before: {Spark, Kafka, Tableau} x {AWS, GCP}

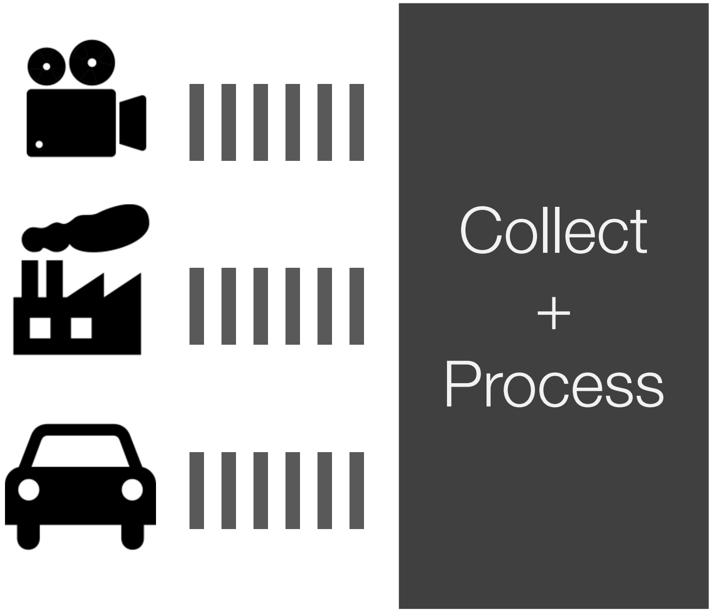
Microsoft, Facebook, Twitter, LinkedIn collect **12M+ events/sec** today

Current Monitoring Pipelines

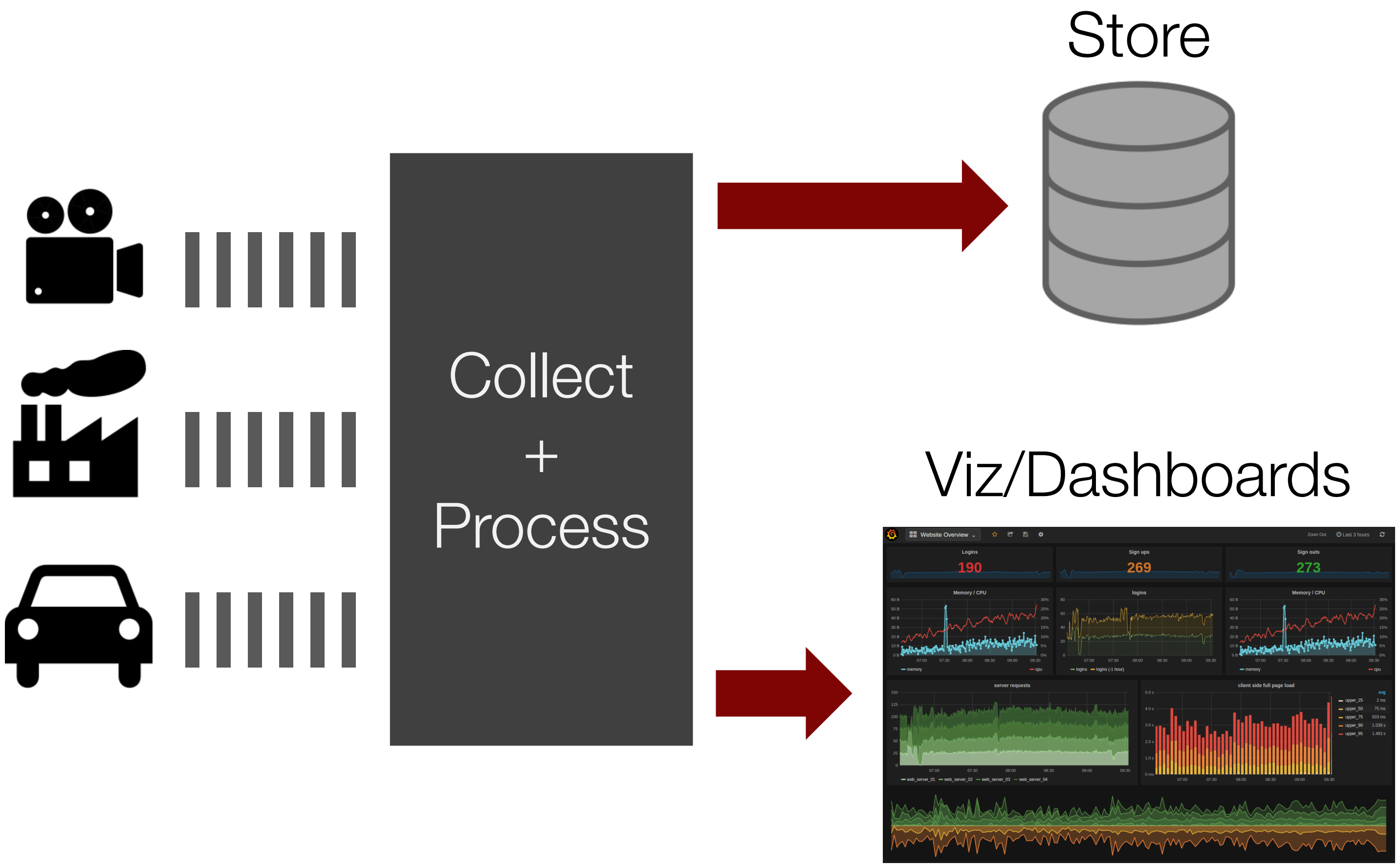
Current Monitoring Pipelines



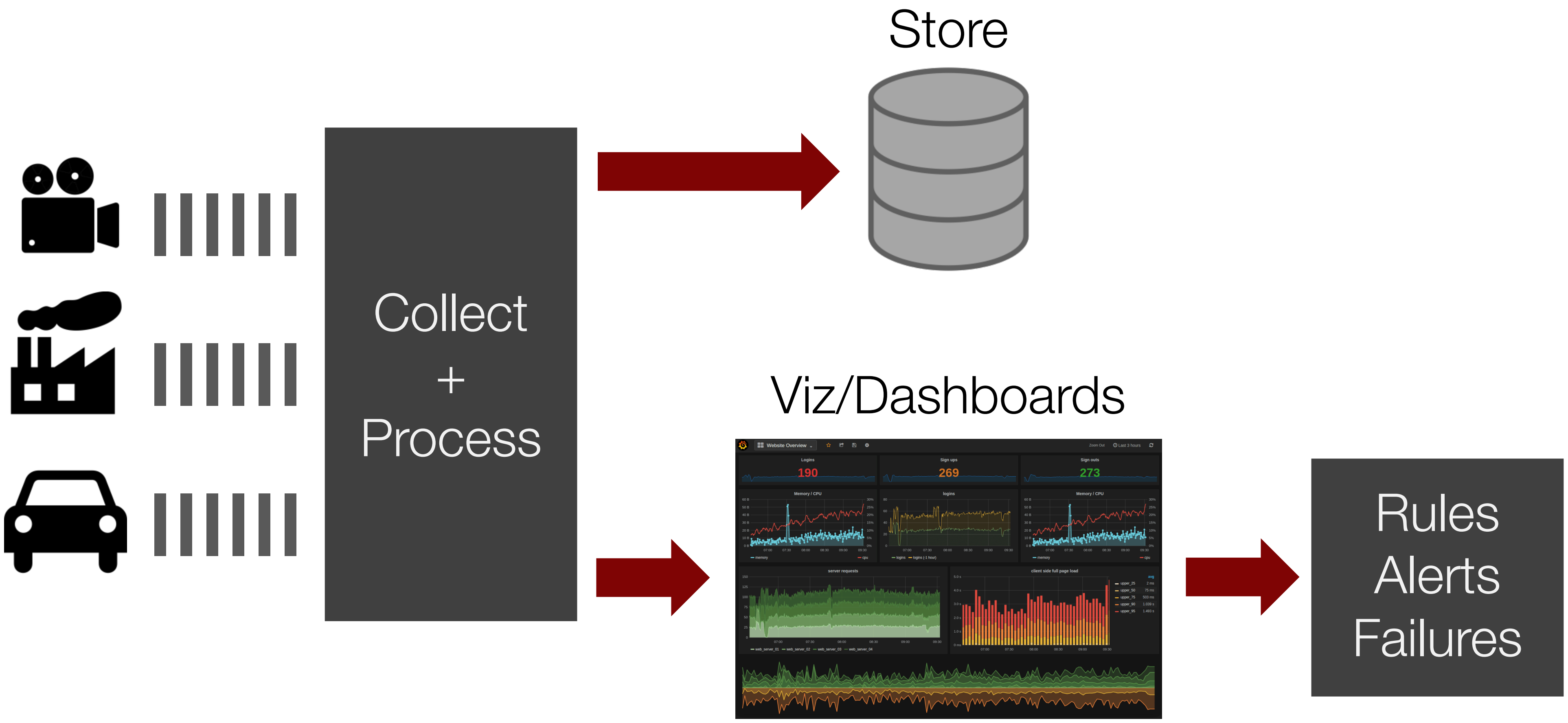
Current Monitoring Pipelines



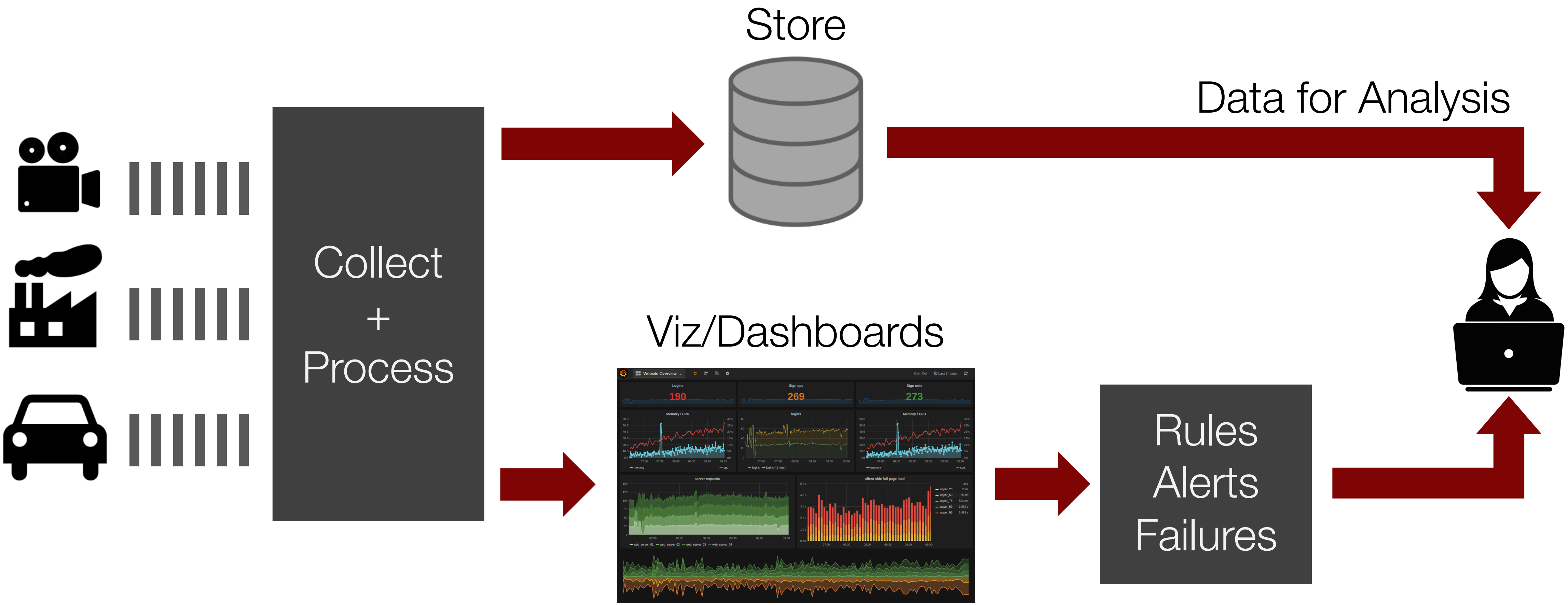
Current Monitoring Pipelines



Current Monitoring Pipelines

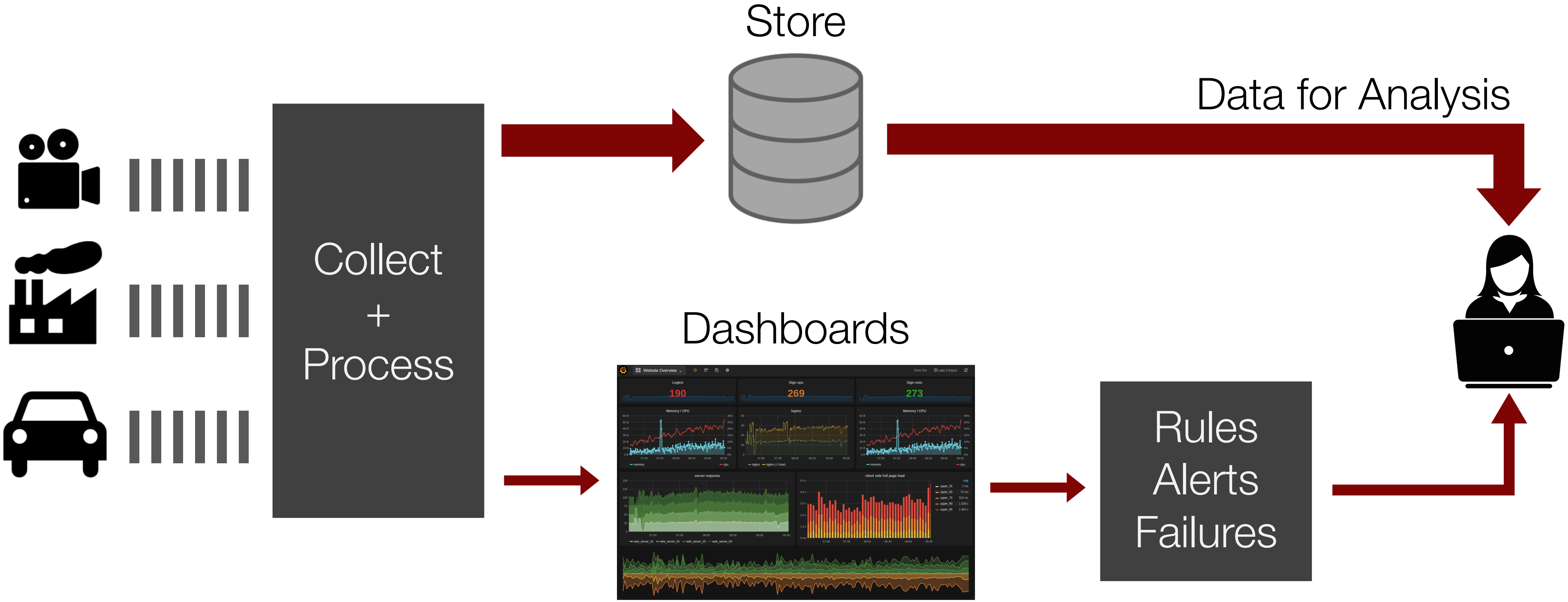


Current Monitoring Pipelines



Current Monitoring Pipelines

Top SV orgs: < 6% data read!



12+ m events/sec

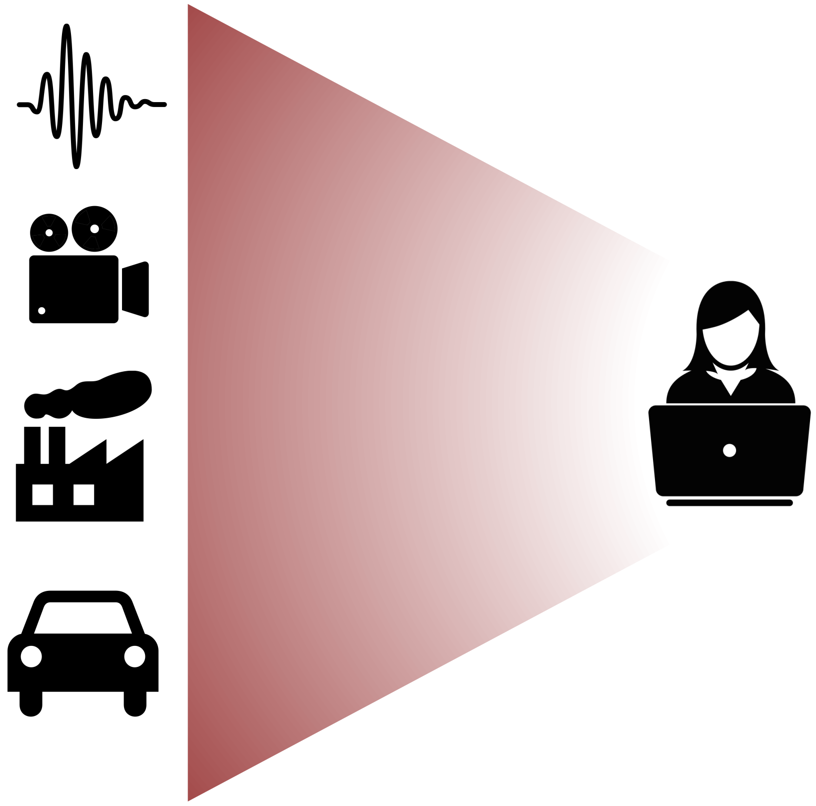
12+ m events/sec

6%

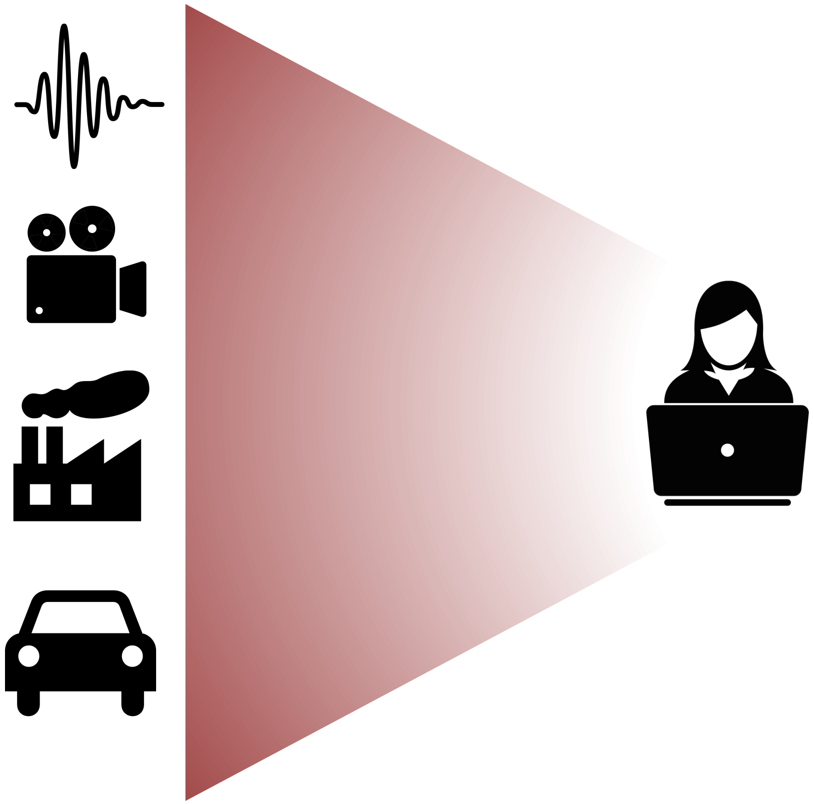
Our research:

How can big-data systems monitor and analyze data more effectively at scale?

Key Bottleneck in Monitoring: Human Attention

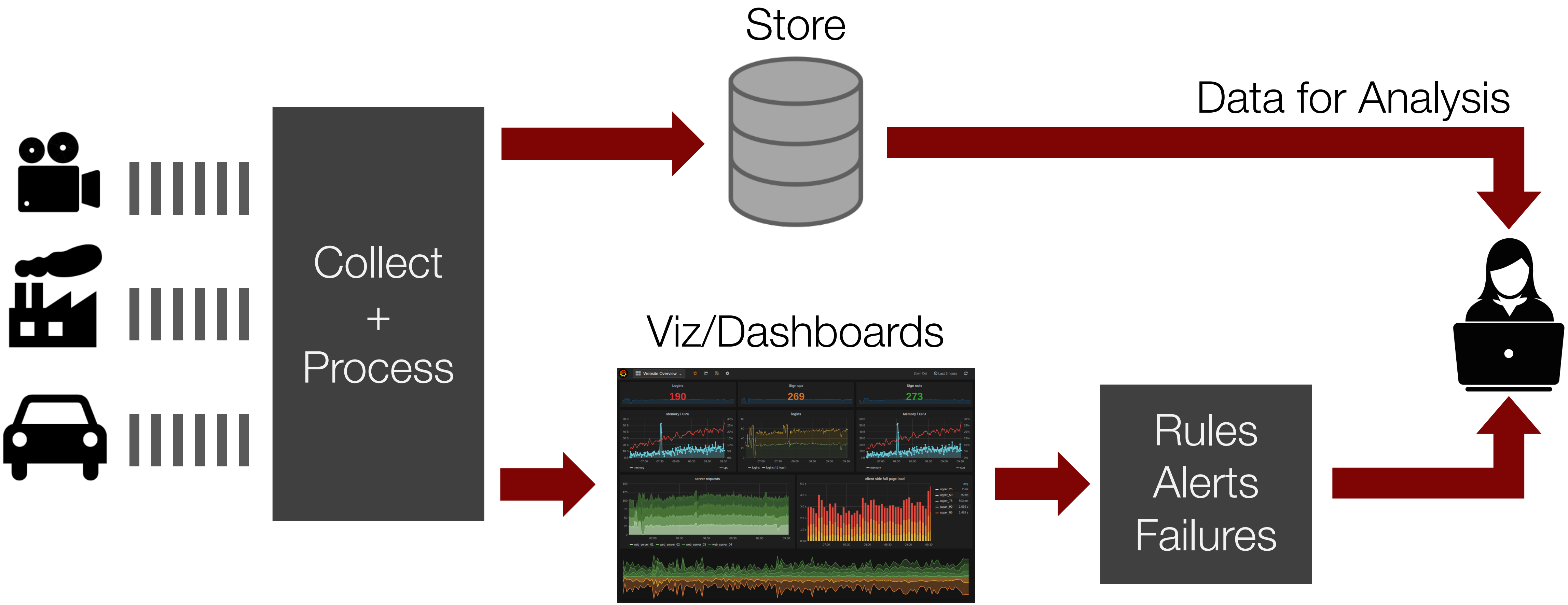


Key Bottleneck in Monitoring: Human Attention



Human attention is scarce!
Infeasible to manually inspect large volumes

Current Monitoring Pipelines



Current Monitoring Pipelines

Key Bottleneck in Monitoring: Human Attention

APACHE HBASE
APACHE STORM™
kafka
APACHE Spark™
ALLUXIO
Spark Streaming
cassandra
Grafana
beam
graphite
Prometheus
APACHE nifi
InfluxDB

Dataflow engines provide a means of processing this data…

…but don't tell us what to show to humans, or what functions to run!

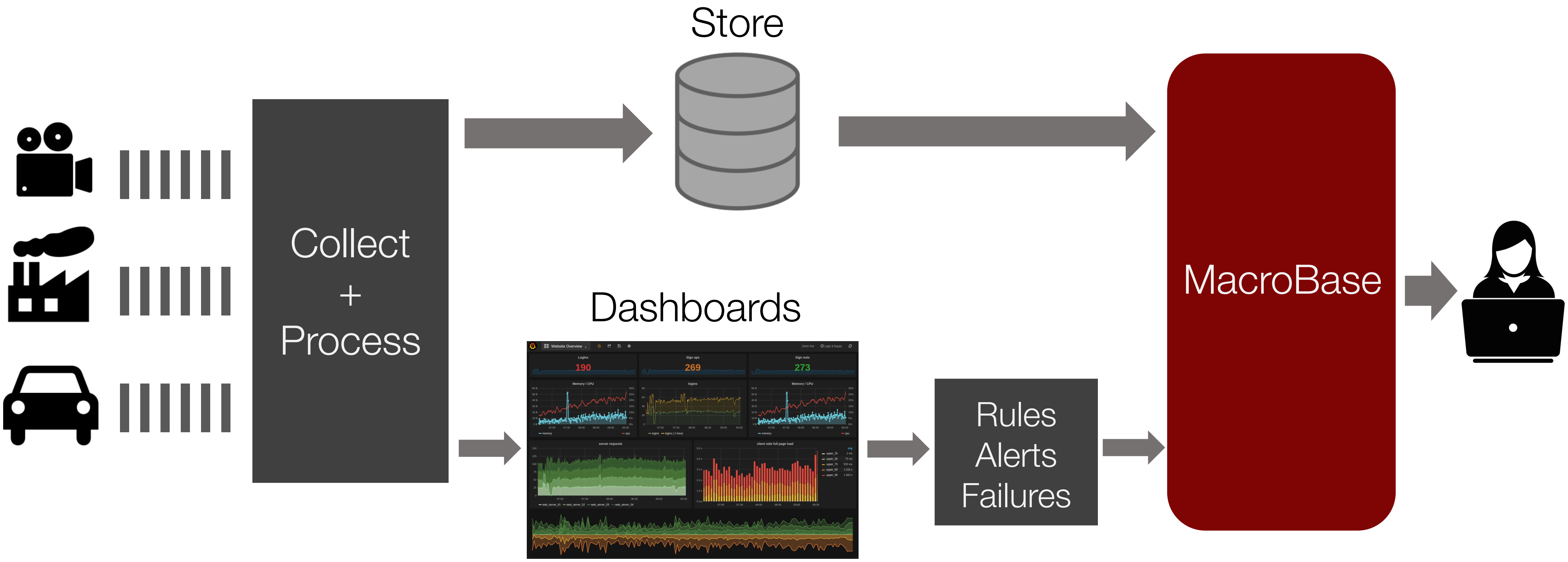
Key Bottleneck in Monitoring: Human Attention

Dataflow engines provide a means of processing this data…

…but don't tell us what to show to humans, or what functions to run!

Stats+ML offer possibilities, but little tried + battle-tested at scale

Monitoring with MacroBase



MacroBase: an analytics engine that **prioritizes user attention** for effective monitoring of high-volume, high-dimensional data

This talk:

Share our goals, architecture, results, and future roadmap

Outline

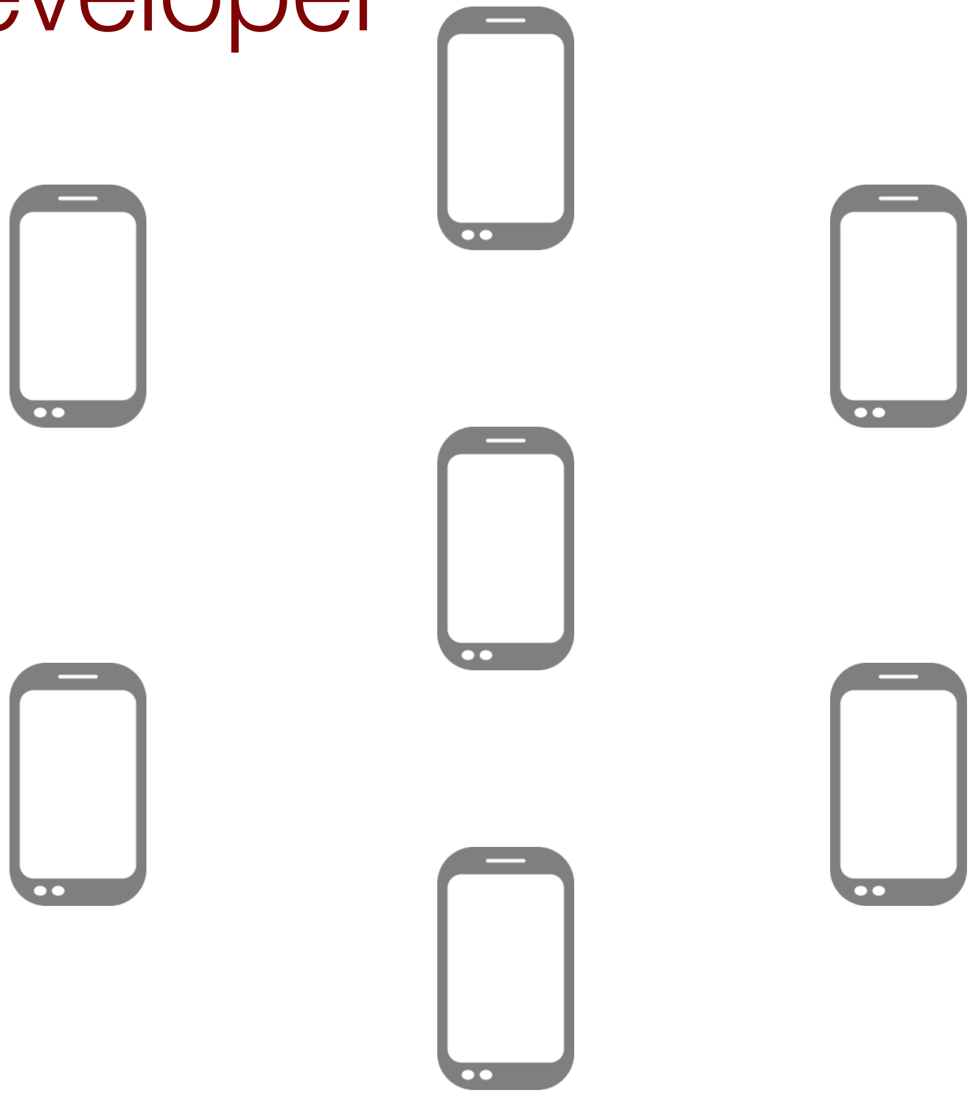
Prioritizing Attention in Fast Data

Demo

Architecture + Usage

A Relational Algebra for MacroBase

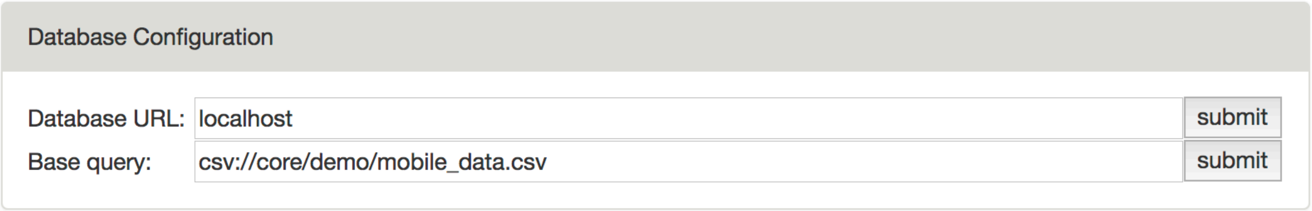
Demo: Mobile App Developer



Demo: Mobile App Developer

Demo: UI Recap

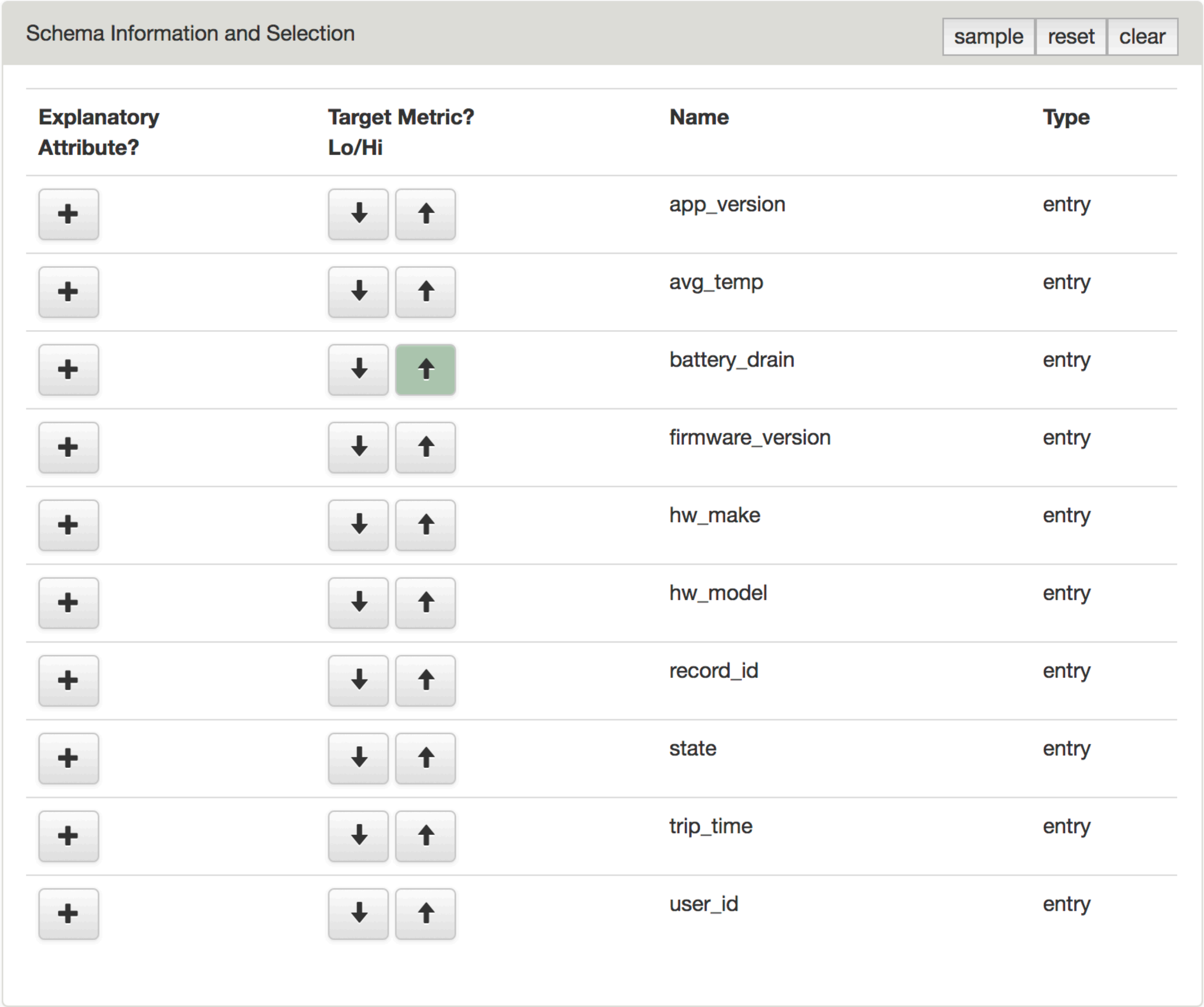
Input data



Demo: UI Recap

Input data

Select metrics

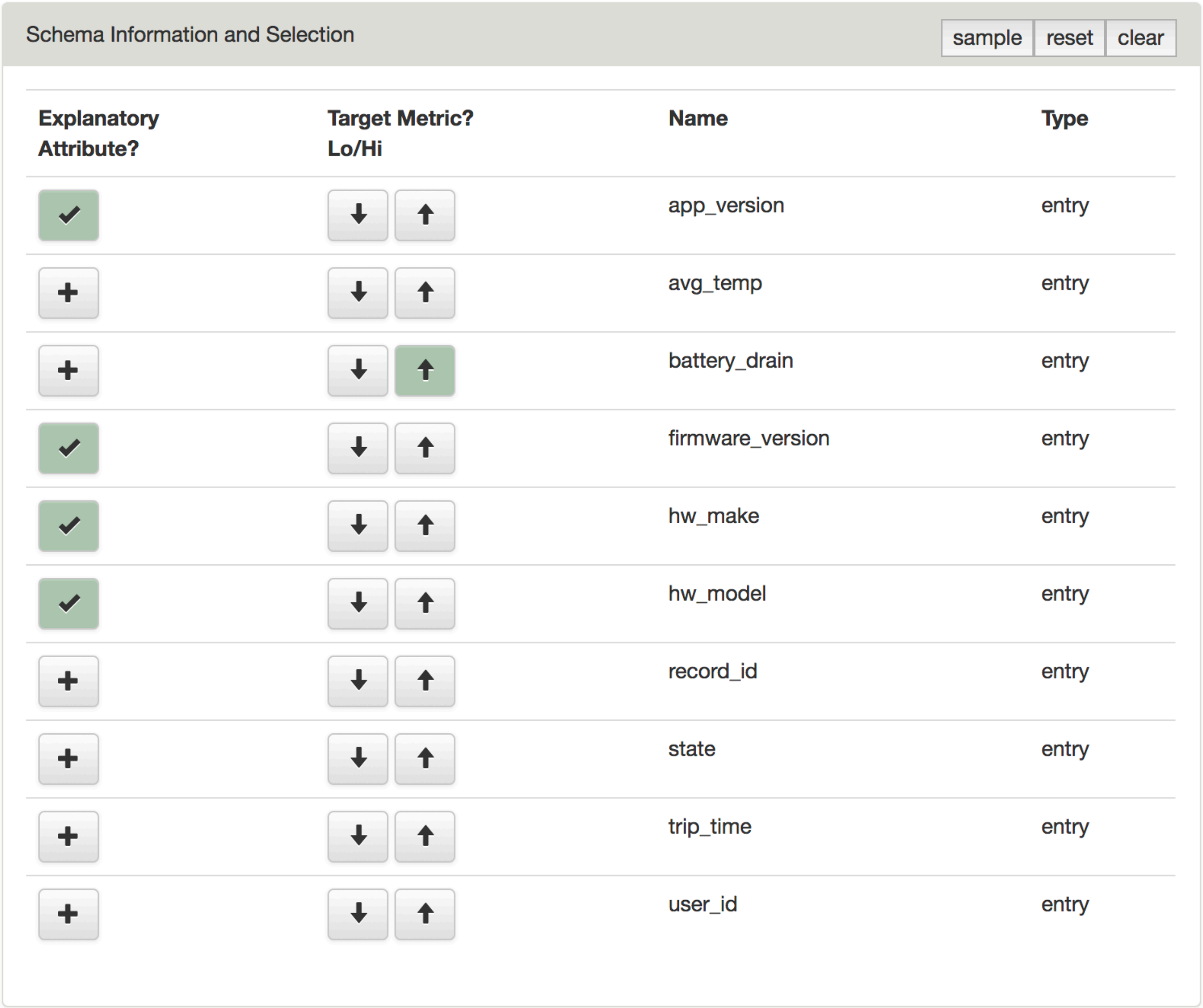


Demo: UI Recap

Input data

Select metrics

Select attributes



Demo: UI Recap

Input data

Select metrics

Select attributes

Explore results

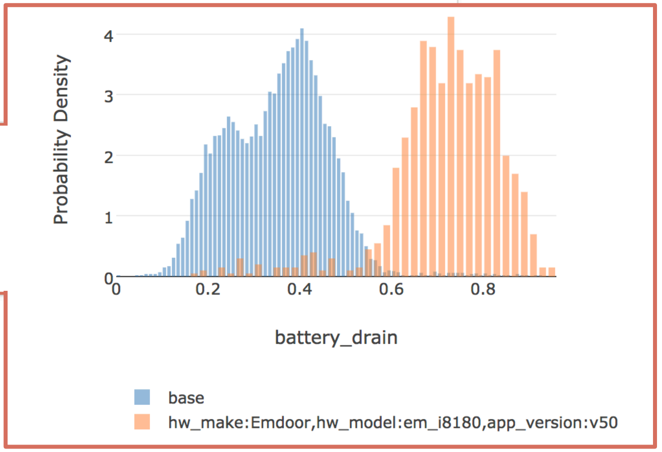
Results

# Outliers	1032
# Inliers	102214
Timing	
Load time	391ms
Processing time	23ms
Summarization time	70ms

Overview Plot: battery_drain

Sort order: support ratio number of attributes

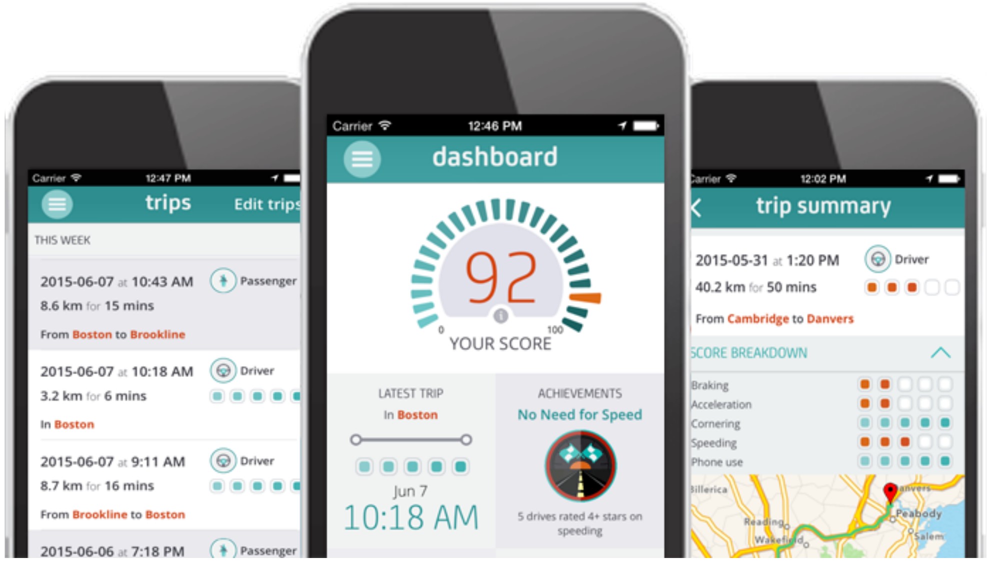
Attributes:	hw_make, hw_model, app_version
Ratio Out/In:	472.89409666285565
Support:	0.8226744186046512
Records:	849



Case Study: CMT

Cambridge Mobile Telematics:

Monitors driving behavior via mobile application available for smartphones

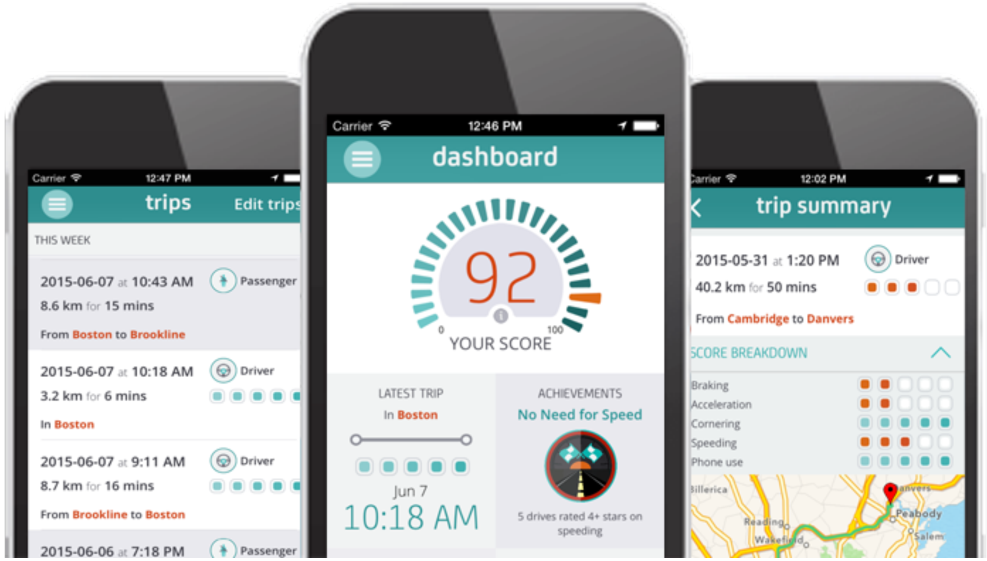


Case Study: CMT

Cambridge Mobile Telematics:

Monitors driving behavior via mobile application available for smartphones

Question: Is the application behaving correctly on every platform?

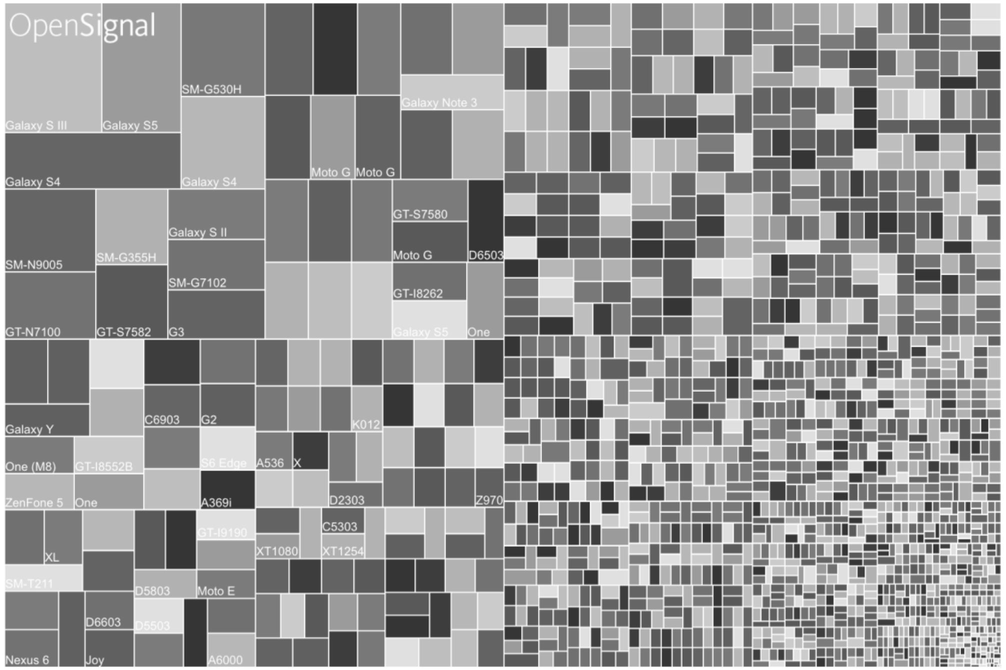


Case Study: CMT

Cambridge Mobile Telematics:

Monitors driving behavior via mobile application available for smartphones

Challenge: Spending even 1 second per deployment combination requires 7 days



25 Major API Releases

Over 24K Android device types

Case Study: CMT

Cambridge Mobile Telematics:

Monitors driving behavior via mobile application available for smartphones

Challenge: Spending even 1 second per deployment combination requires 7 days

"iOS 9.0 beta 1–5 (but not 9.0.1) had a buggy Bluetooth stack that prevented iOS devices from connecting to CMT devices."

Outline

Prioritizing Attention in Fast Data

Demo

Architecture + Usage

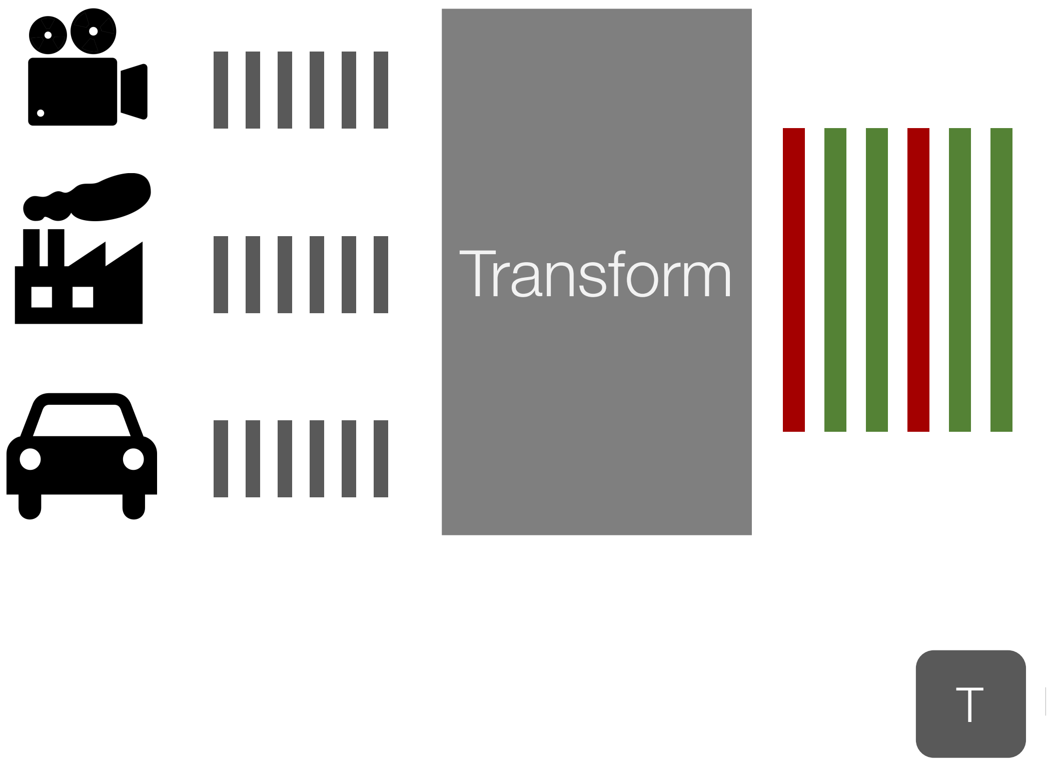
A Relational Algebra for MacroBase

MacroBase Architecture: Operator Cascades

Execute operator cascades to transform, segment, and explain streams

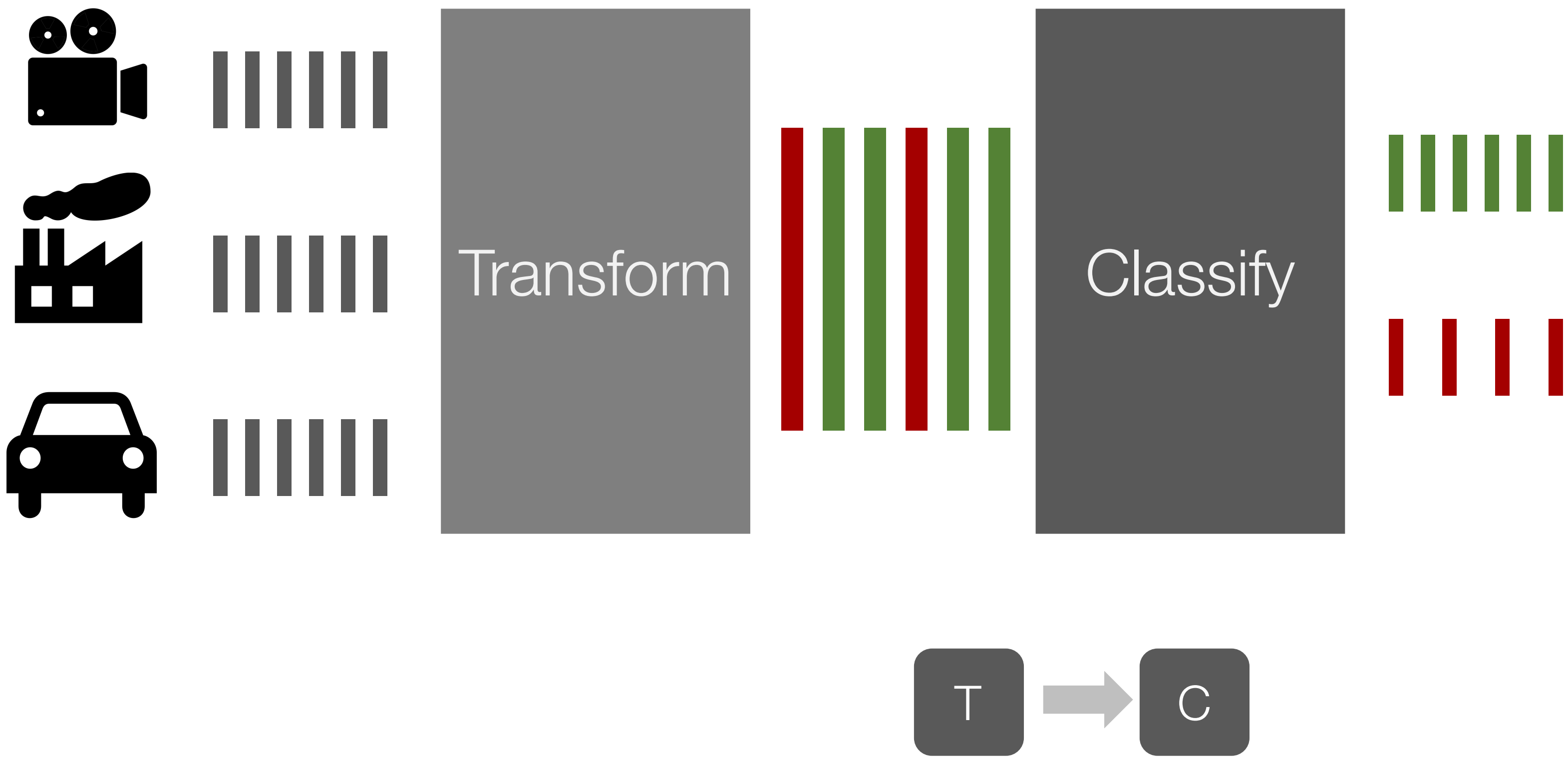
MacroBase Architecture: Operator Cascades

Execute operator cascades to transform, segment, aggregate streams

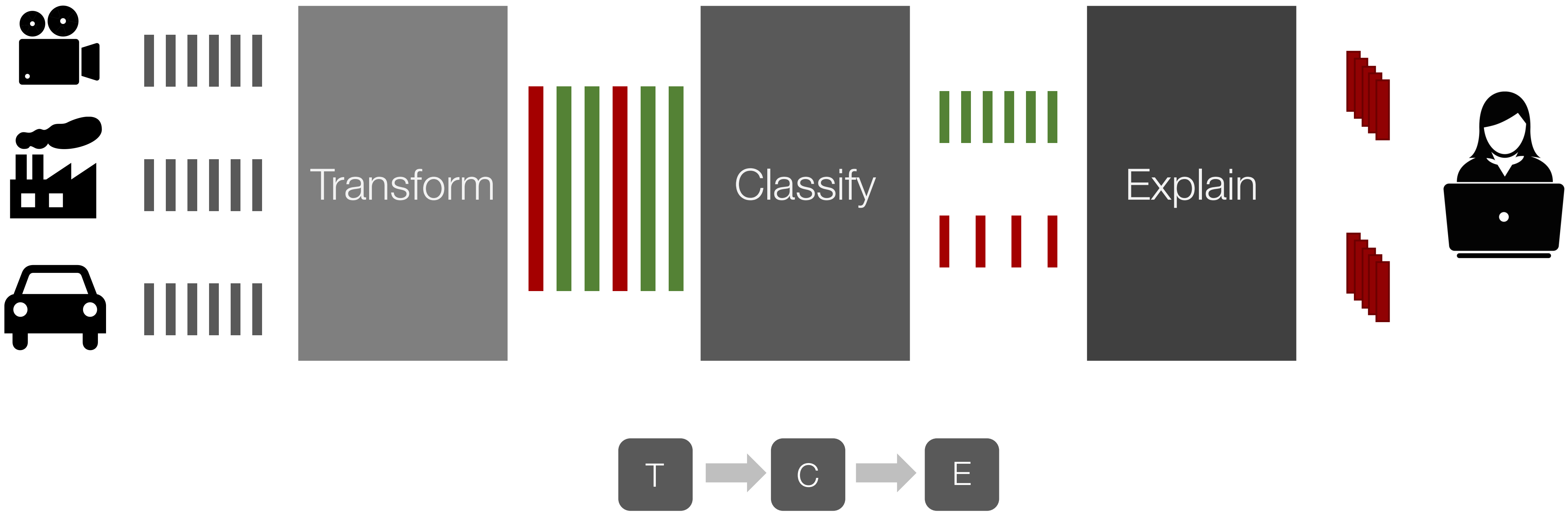


MacroBase Architecture: Operator Cascades

Execute operator cascades to transform, segment, aggregate streams

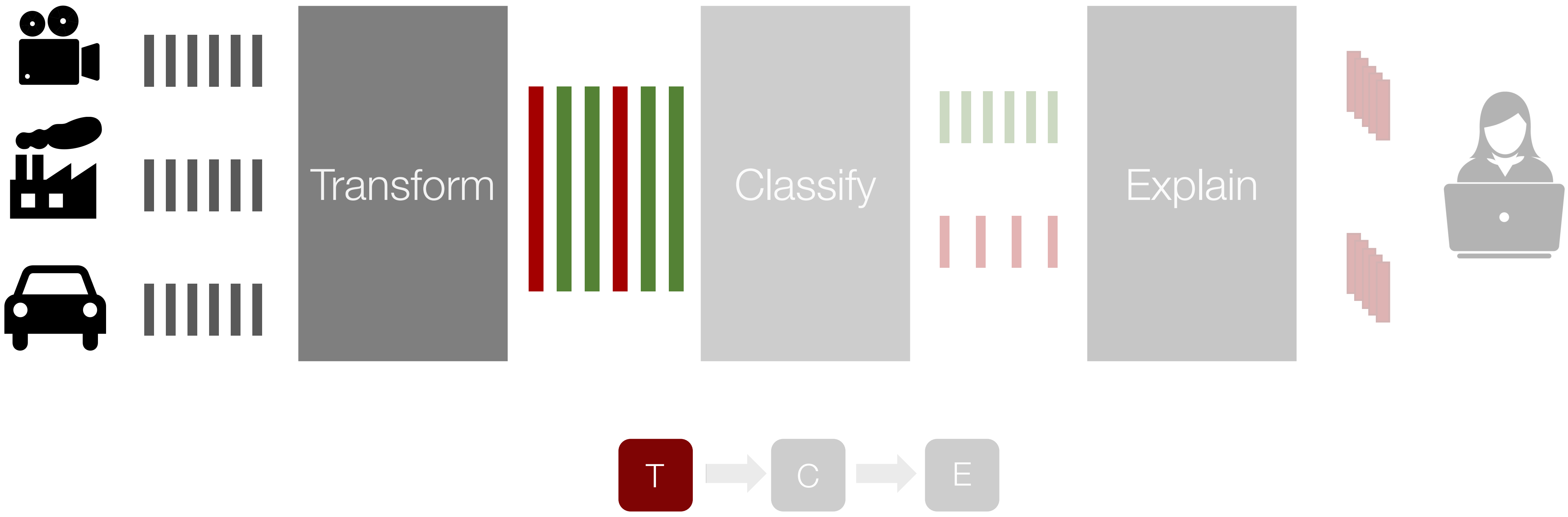


MacroBase Architecture: Operator Cascades

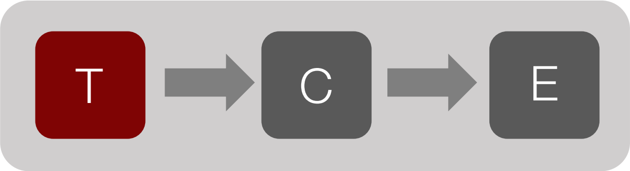


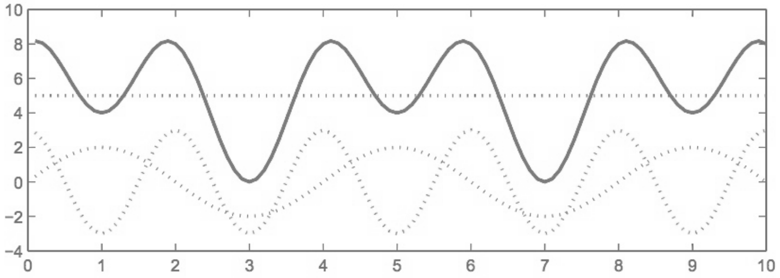
Transformation

Feature extraction, dimensionality reduction, streaming ETL

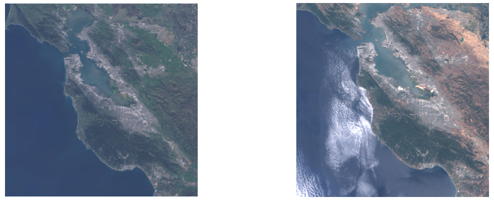


Transformation





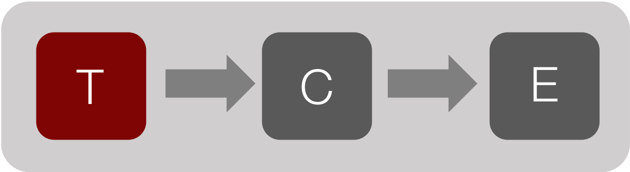
e.g., time series dimensionality reduction (via FFT, PCA)

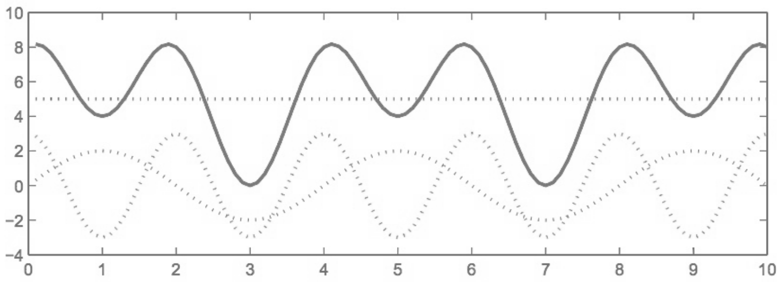


e.g., image-specific features (e.g., hue and luminosity)

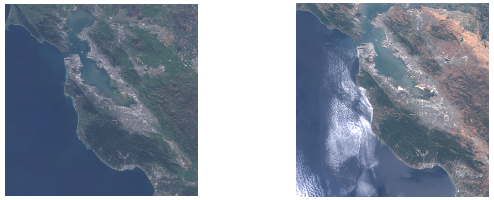
Optional

Transformation





e.g., time series dimensionality reduction (via FFT, PCA)

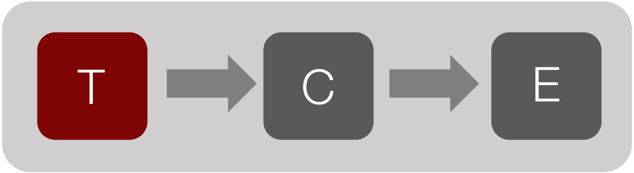


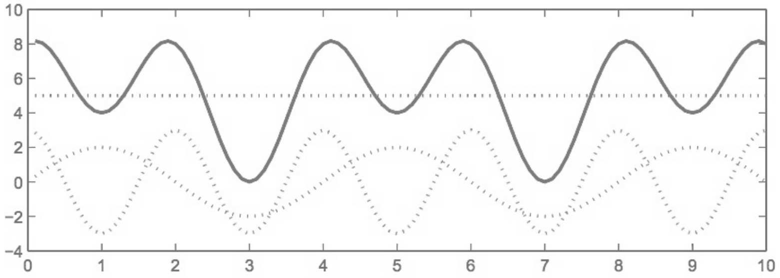
e.g., image-specific features (e.g., hue and luminosity)

Optional

Domain-specific data pre-processing

Transformation





e.g., time series dimensionality reduction (via FFT, PCA)

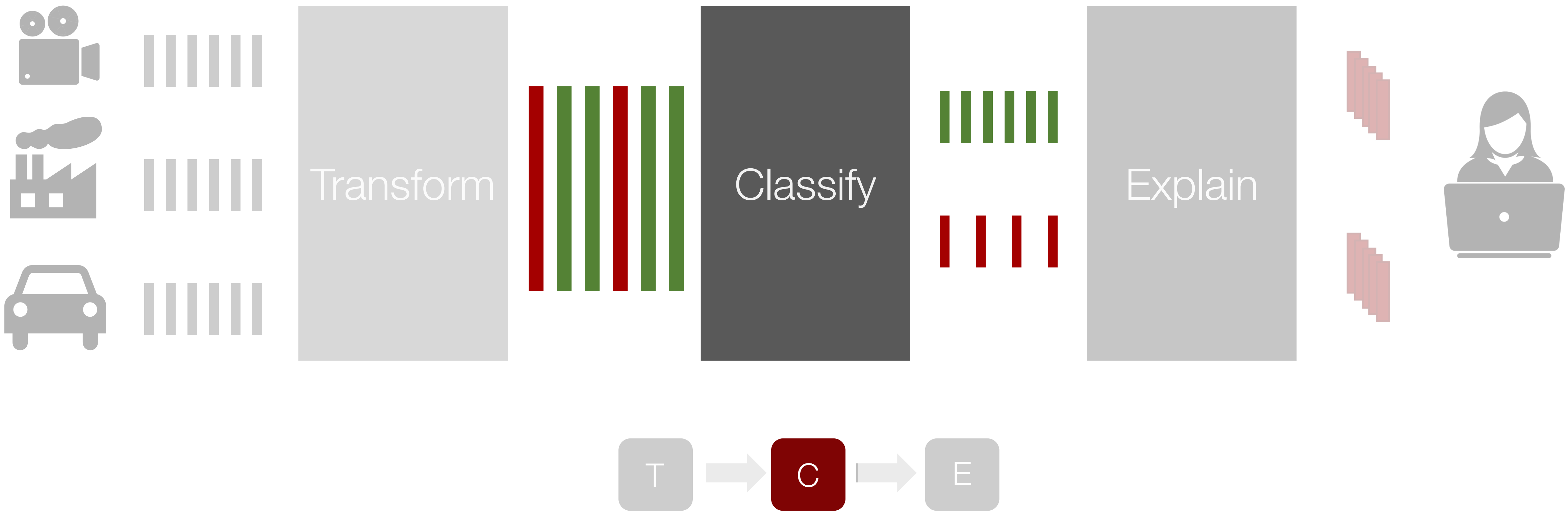
e.g., image-specific features (e.g., hue and luminosity)

Domain-specific data pre-processing

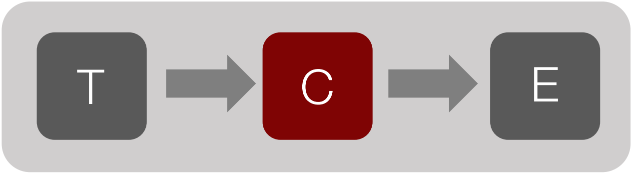
Combine and chain transformations to build complex features

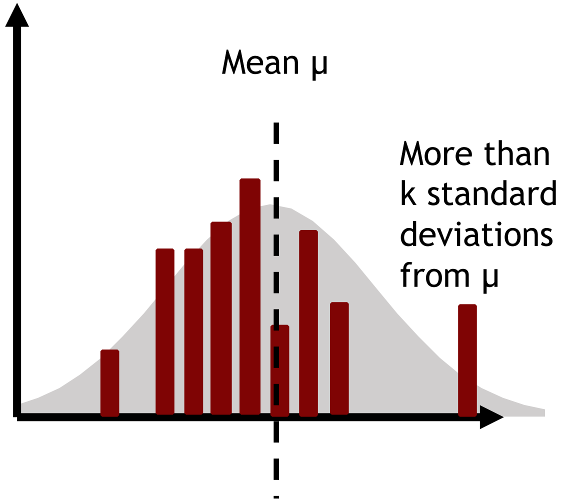
Classification

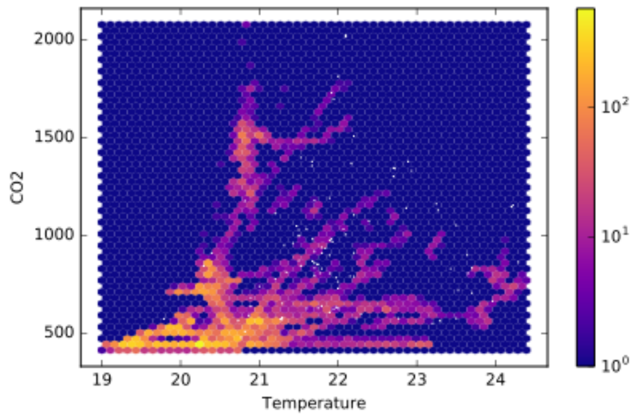
Segmentation, rule evaluation, data filtering



Classification



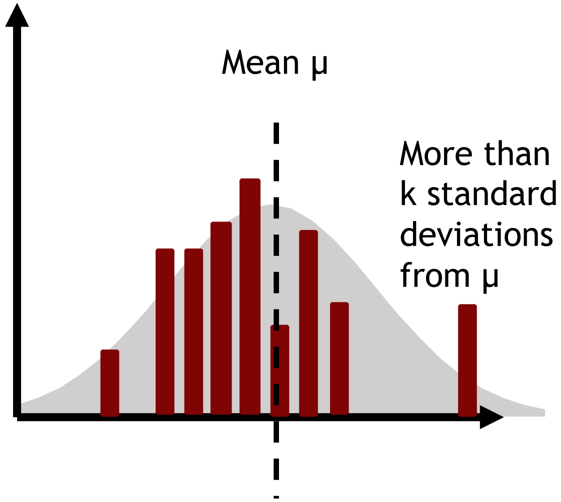


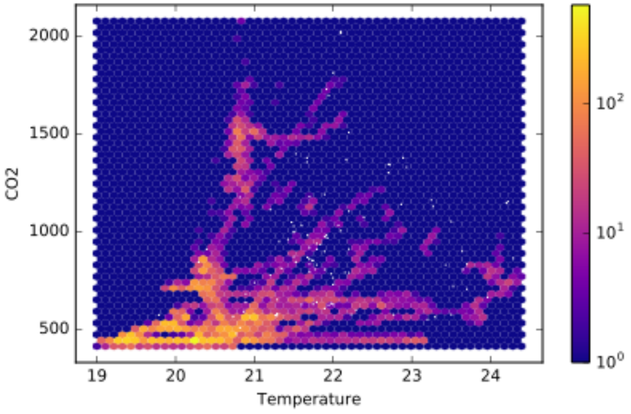


Segment and filter stream by target behavior (e.g., abnormalities)

Classification

T → C → E

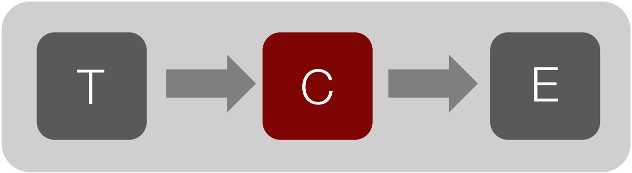


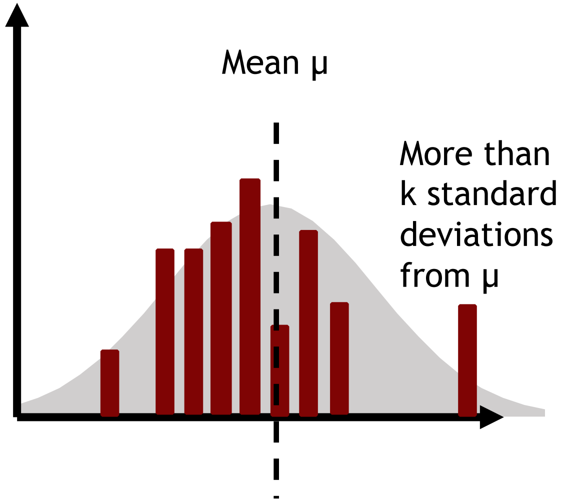


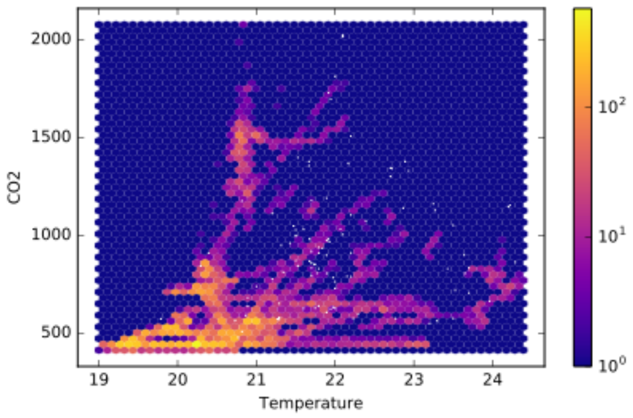
Segment and filter stream by target behavior (e.g., abnormalities)

Default: identify unlikely data points (e.g., via density estimation)

Classification







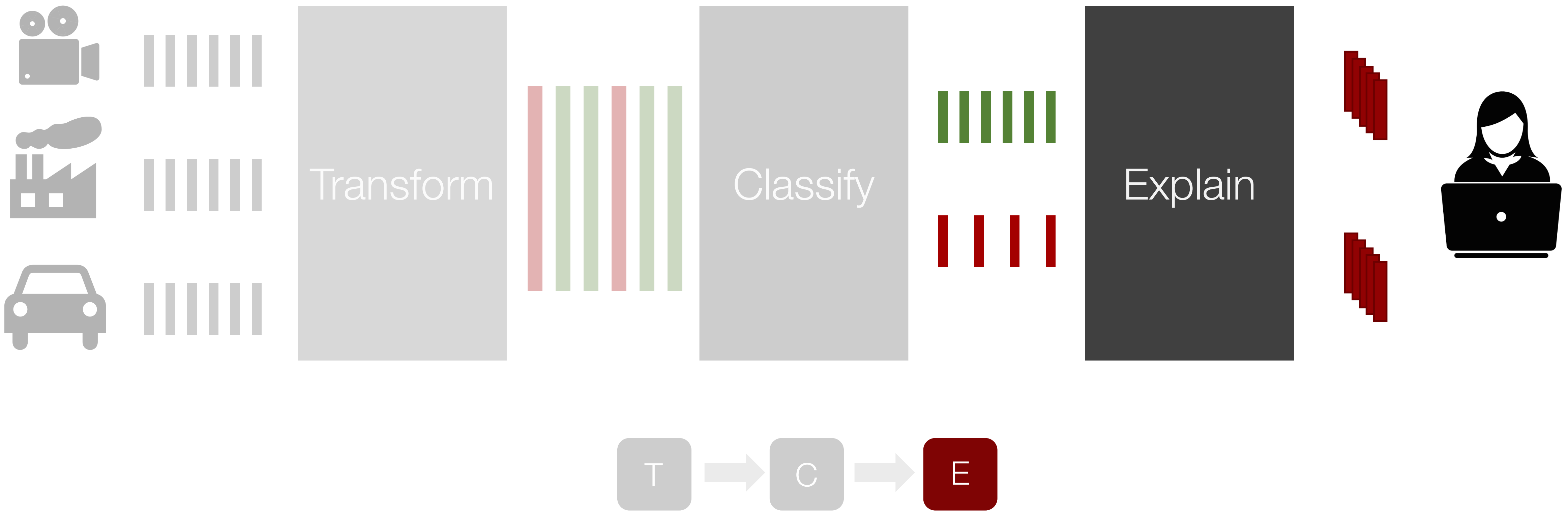
Segment and filter stream by target behavior (e.g., abnormalities)

Default: identify unlikely data points (e.g., via density estimation)

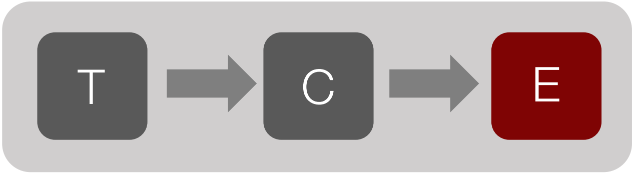
Combine with thresholds, predicates, or custom classifiers

Explanation

Find underlying causes for classified abnormalities



Explanation



Errors

{iPhone7, Canada}
{iPhone7, USA}
{iPhone8, Canada}
{iPhone7, USA}
{iPhone8, Canada}

Canada may have a problem!

Non-Errors

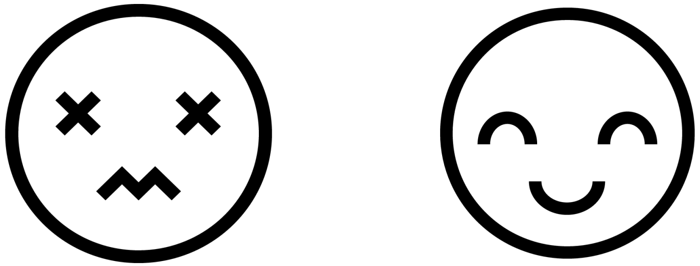
{iPhone8, USA}
{iPhone7, USA}
{iPhoneX, USA}
{iPhone7, USA}
{iPhone7, USA}
{iPhone8, USA}
{iPhone7, USA}
{iPhone7, USA}

Explain classification results by identifying behavior correlated with being filtered

Explanation

T → C → E

Relative Risk Ratio



error non-error

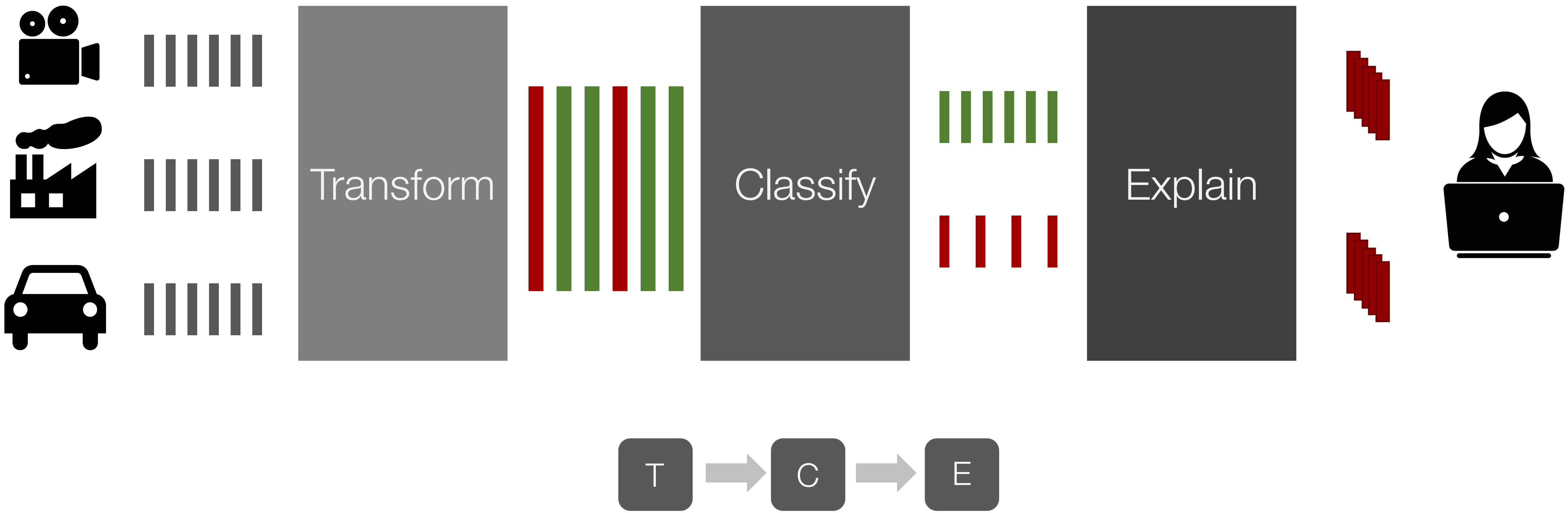
Explain classification results by identifying behavior correlated with being filtered

Default: relative risk ratio based on data attributes

$$\frac{P(\text{error} \mid \text{Canada})}{P(\text{error} \mid \text{not Canada})} = \frac{\frac{\#\text{ outliers w/ Canada}}{\#\text{ tuples w/ Canada}}}{\frac{\#\text{ outliers w/out Canada}}{\#\text{ tuples w/out Canada}}} = \frac{\frac{3}{3}}{\frac{2}{10}}$$

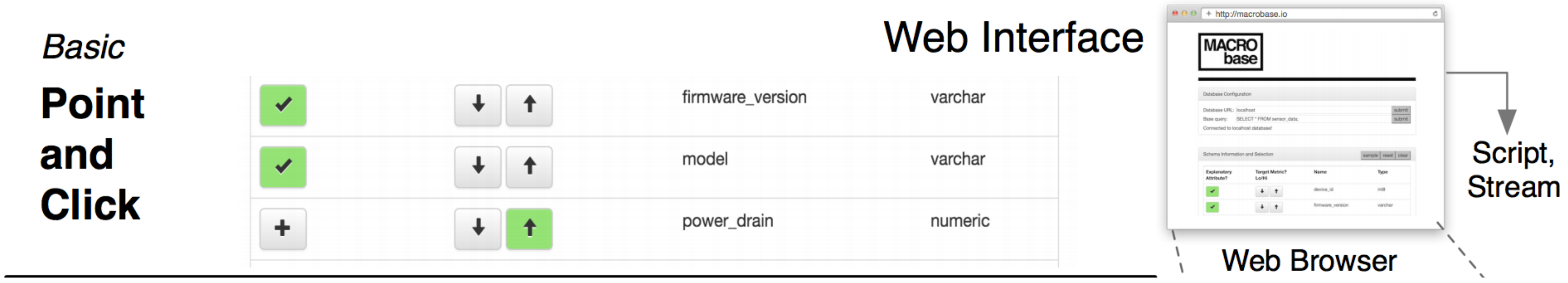
MacroBase Architecture: Operator Cascades

Execute operator cascades to transform, segment, aggregate streams



Usage

Usage



Basic
Point and Click
firmware_version
varchar
model
varchar
power_drain
numeric
Web Interface
http://macrobase.io
MACRO base
Script, Stream
Web Browser

Usage

Basic

Point and Click

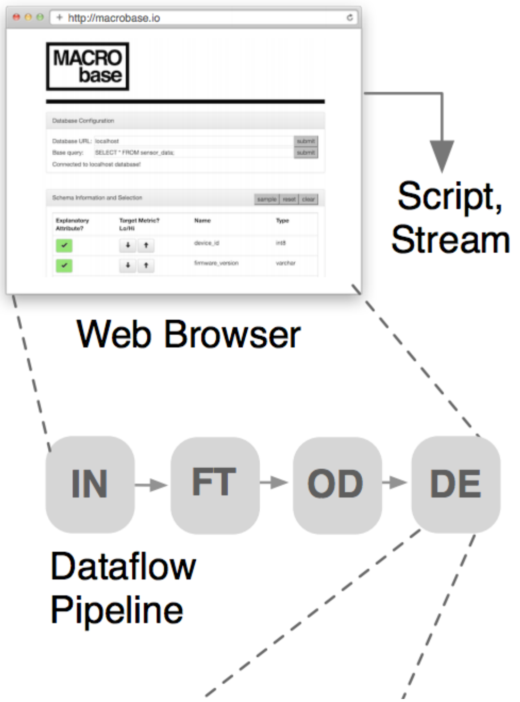
Web Interface

Intermediate

Custom Pipeline Config

Java

```
new LinearMetricNormalizer()
        .then(new MBGroupBy(groupByIndex,
                    () -> new FeatureTransform(conf)))
        .then(new BatchingPercentileClassifier(conf))
        .then(new BatchSummarizer(conf))
       .consume(conf.constructIngester().getStream().drain());
```



Usage

Basic

Point and Click

Web Interface

Intermediate

Custom Pipeline Config

Java

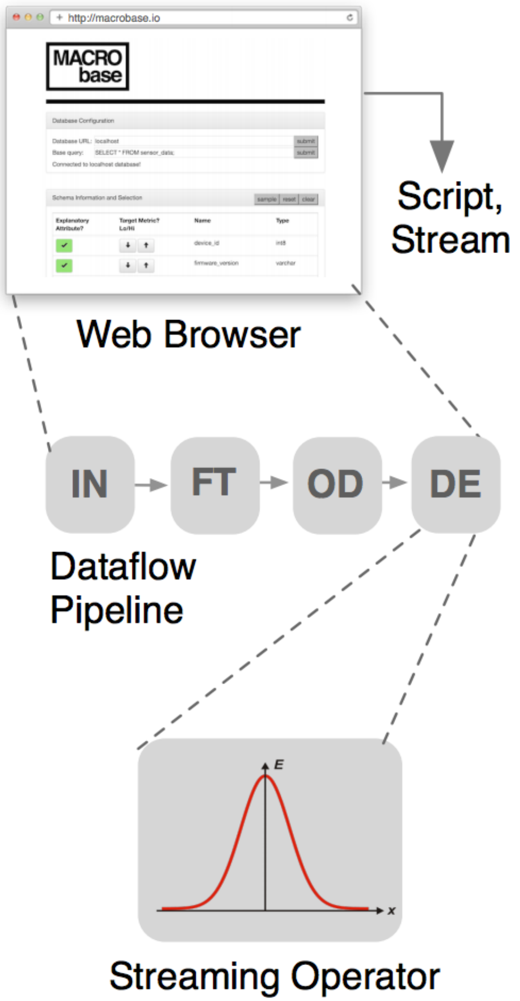
```
new LinearMetricNormalizer()
        .then(new MBGroupBy(groupByIndex,
                    () -> new FeatureTransform(conf)))
        .then(new BatchingPercentileClassifier(conf))
        .then(new BatchSummarizer(conf))
       .consume(conf.constructIngester().getStream().drain());
```

Advanced

Custom Dataflow Operators

Java / C++

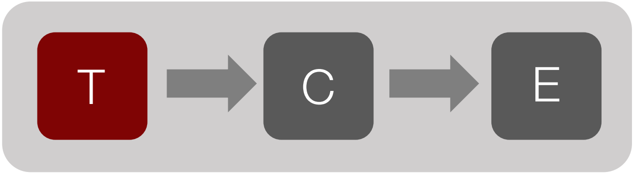
```
int k = data.get(0).metrics().getDimension();
int n = data.size();
List<double[]> metrics = new ArrayList<>(n);
for (Datum curDatum : data)
        metrics.add(curDatum.metrics().toArray());
List<double[]> trimmedMetrics = trimmer.process(metrics);
gModel = new Gaussian().fit(trimmedMetrics);
```



Usage—JSON Rest API

```
{
 "inputURI": ...,
 "metric": "Percentile Dropped records",
 "classifier": "quantile",
 "cutoff": 1.0,
 "includeHi": true,
 "includeLo": false,
 "attributes": ["SDK Version", "Network Type", "App Version",
       "OS Version",   "Device"],
 "minSupport": 0.005,
 "minRatioMetric": 1.5
}
```

Online Progress Estimation in Dimensionality Reduction

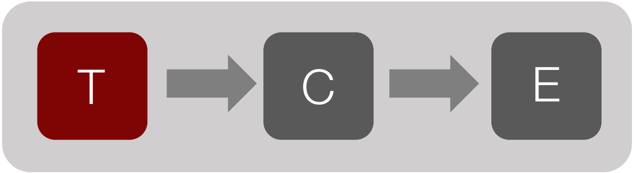


<u>Principal Component Analysis</u>

Core dimensionality reduction operator for many applications

[Suri and Bailis, arXiv 2017]

Online Progress Estimation in Dimensionality Reduction



Principal Component Analysis

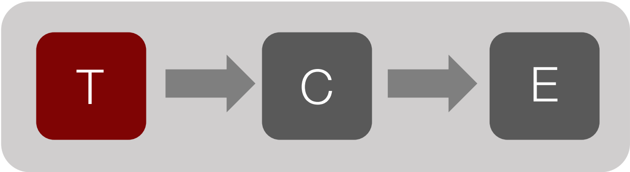
Core dimensionality reduction operator for many applications

Out-of-the-box implementations are extremely slow

$O(\min[mn^2, nm^2])$ via singular value decomposition

[Suri and Bailis, arXiv 2017]

Online Progress Estimation in Dimensionality Reduction



Principal Component Analysis

Core dimensionality reduction operator for many applications

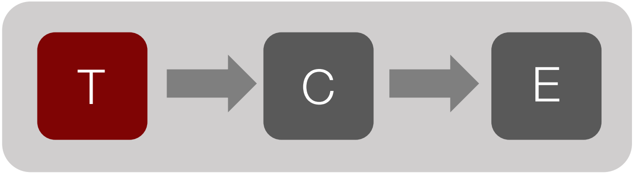
Out-of-the-box implementations are extremely slow

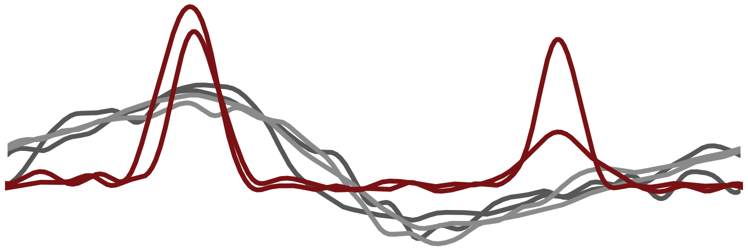
$O(\min[mn^2, nm^2])$ via singular value decomposition

Two insights enable significantly faster performance in practice even with naïve PCA implementations

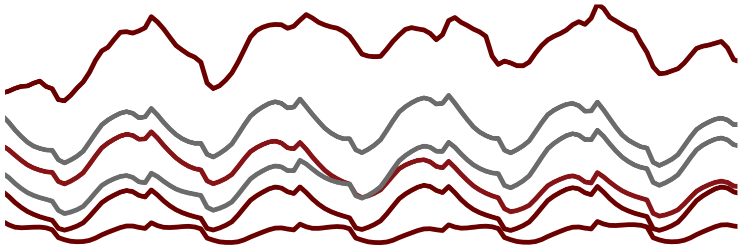
[Suri and Bailis, arXiv 2017]

Online Progress Estimation in Dimensionality Reduction





Variable Star Brightness

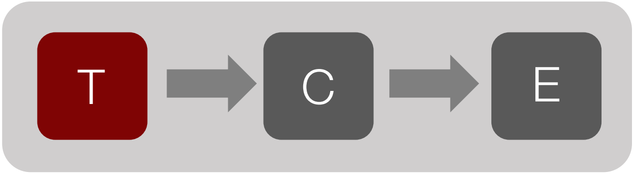


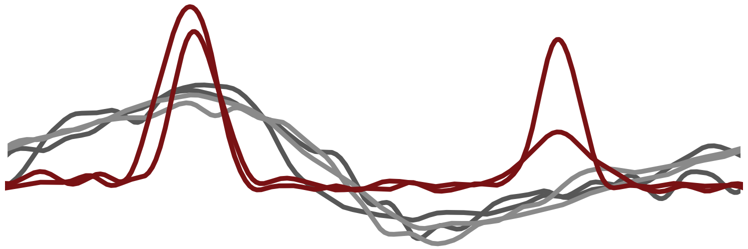
Fan Power Consumption

Data sources are structured; sample prior to model

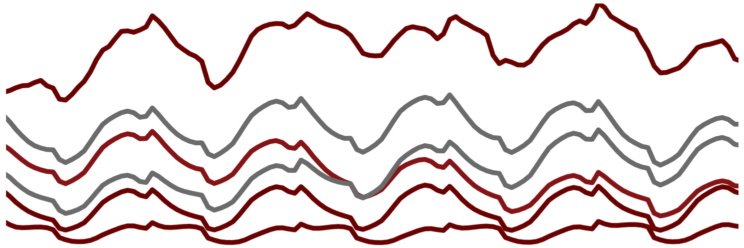
[Suri and Bailis, arXiv 2017]

Online Progress Estimation in Dimensionality Reduction





Variable Star Brightness



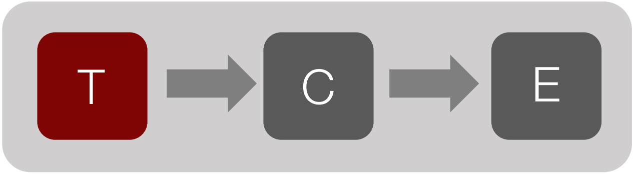
Fan Power Consumption

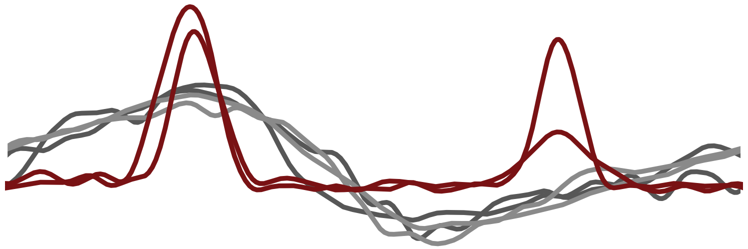
Data sources are structured; sample prior to model

Dimensionality reduction is a pre-processing step; sample until too expensive

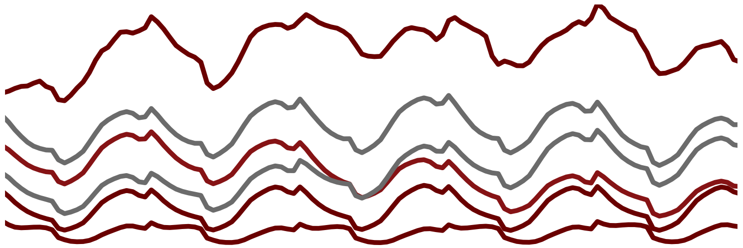
[Suri and Bailis, arXiv 2017]

Online Progress Estimation in Dimensionality Reduction





Variable Star Brightness



Fan Power Consumption

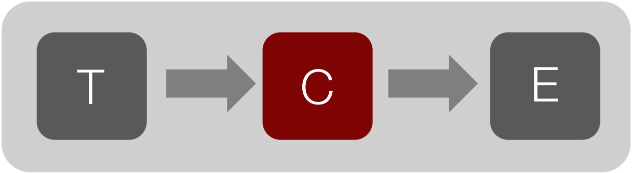
[Suri and Bailis, arXiv 2017]

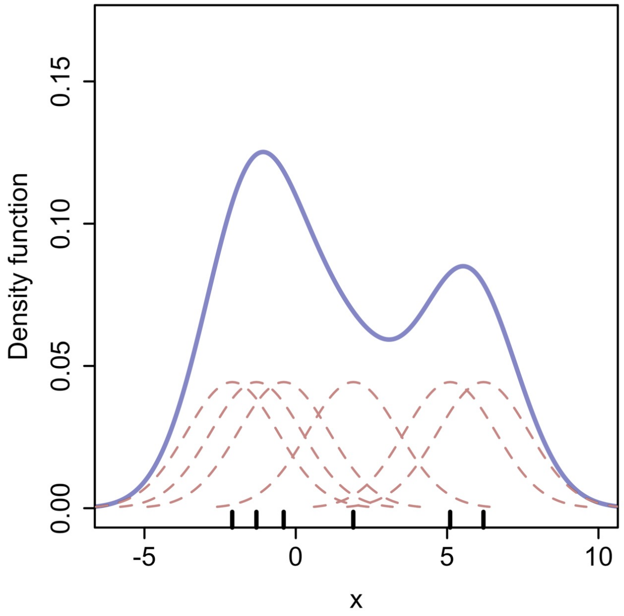
Data sources are structured; sample prior to model

Dimensionality reduction is a pre-processing step; sample until too expensive

50x speedup in dimensionality reduction, and 33x speedup in end-to-end pipelines compared to PCA via SVD

Predicate Pushdown in Density Estimation





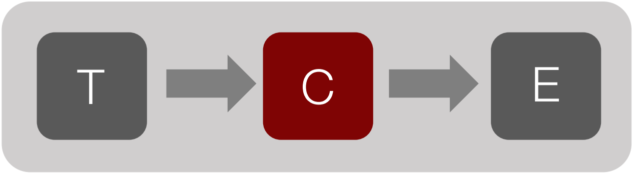
<u>Kernel Density Estimation</u>

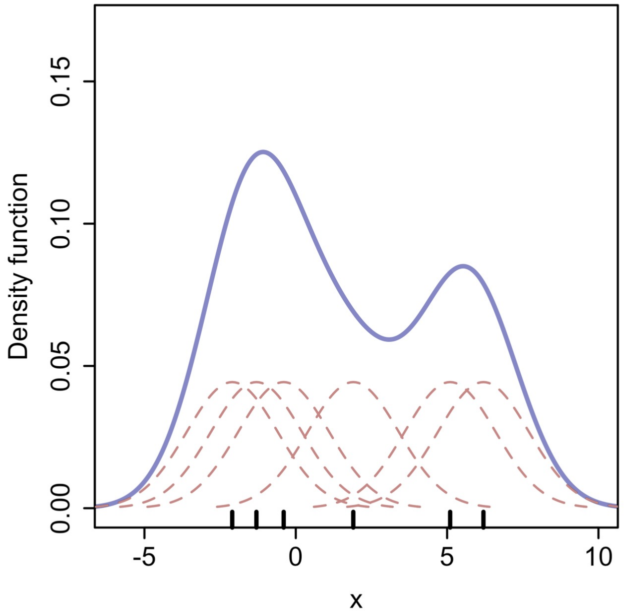
Each point contributes a small “kernel”

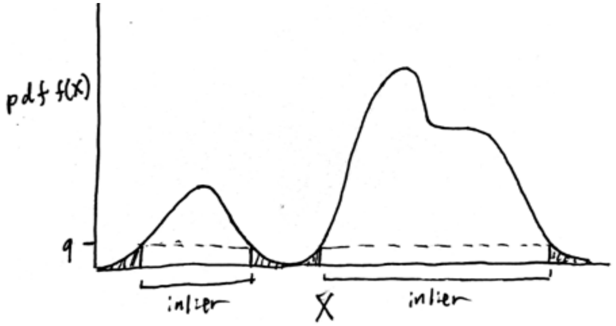
Asymptotically optimal estimation

[Gan and Bailis, SIGMOD 2017]

Predicate Pushdown in Density Estimation







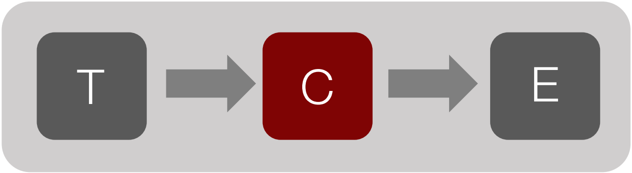
Kernel Density Estimation

Each point contributes a small “kernel”

Asymptotically optimal estimation

[Gan and Bailis, SIGMOD 2017]

Predicate Pushdown in Density Estimation



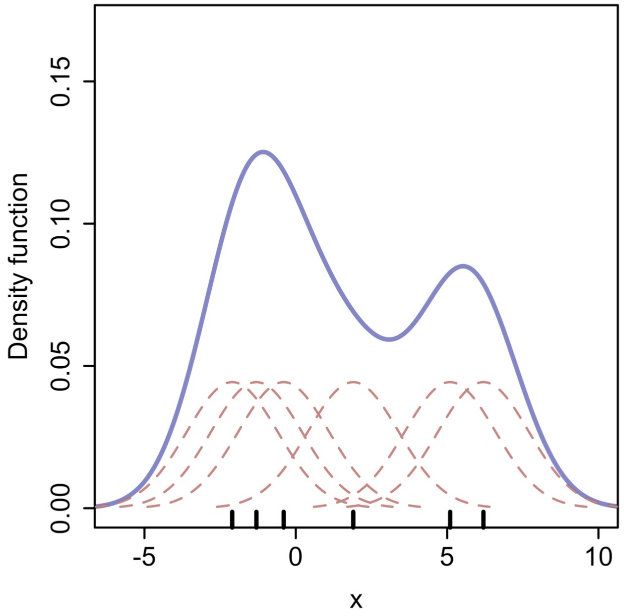
Kernel Density Estimation

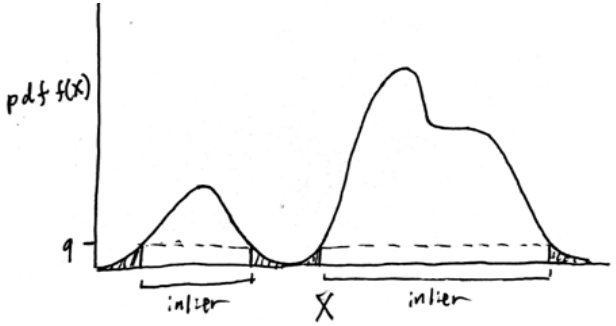
Each point contributes a small “kernel”

Asymptotically optimal estimation

Compute density: $O(n^2)$

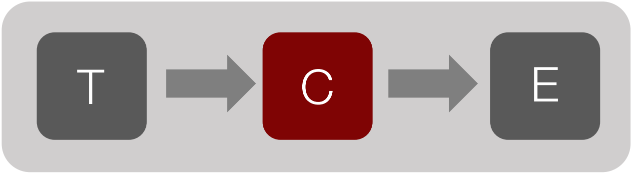
500K points: 2 hours on 2.4GHz CPU!





[Gan and Bailis, SIGMOD 2017]

Predicate Pushdown in Density Estimation



Kernel Density Estimation

Each point contributes a small "kernel"

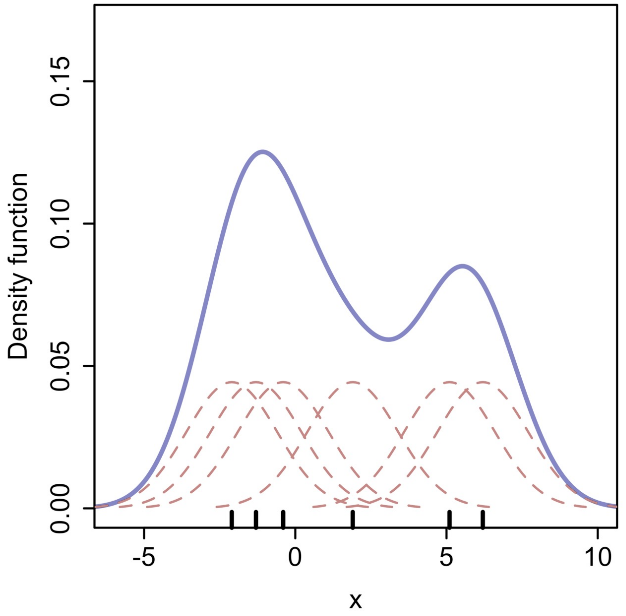
Asymptotically optimal estimation

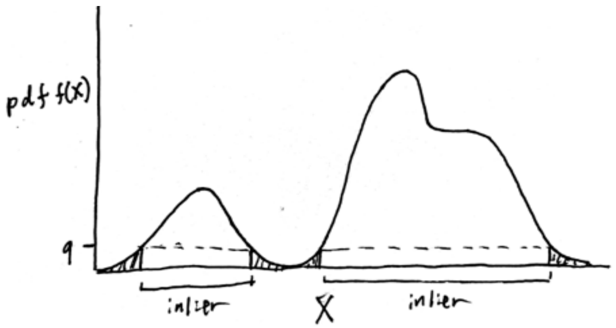
Compute density: $O(n^2)$

500K points: 2 hours on 2.4GHz CPU!

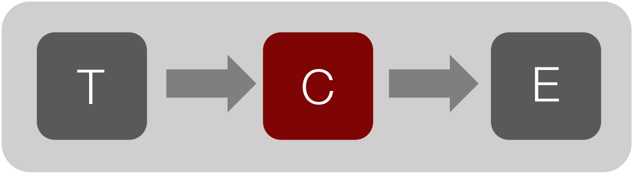
Can we do better?

[Gan and Bailis, SIGMOD 2017]



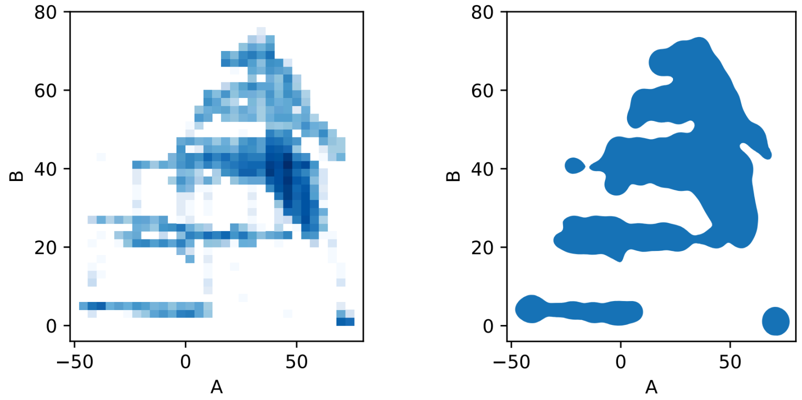


Predicate Pushdown in Density Estimation



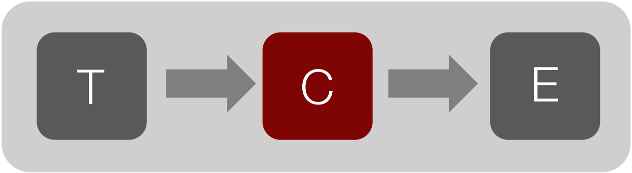
Classification: only need to tell whether above or below target

Don’t need to compute exact densities!



[Gan and Bailis, SIGMOD 2017]

Predicate Pushdown in Density Estimation

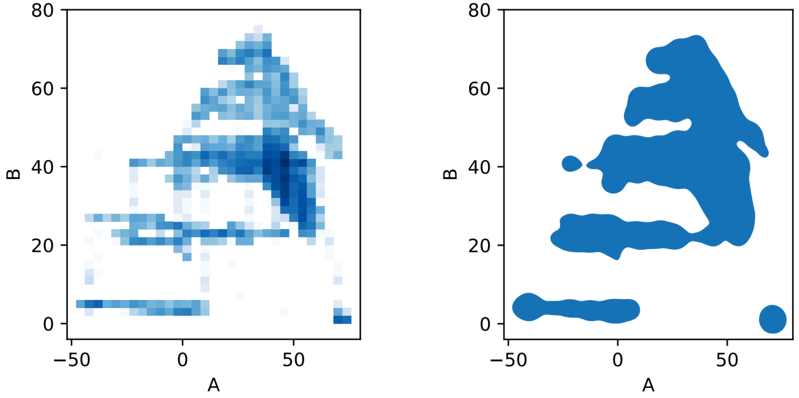


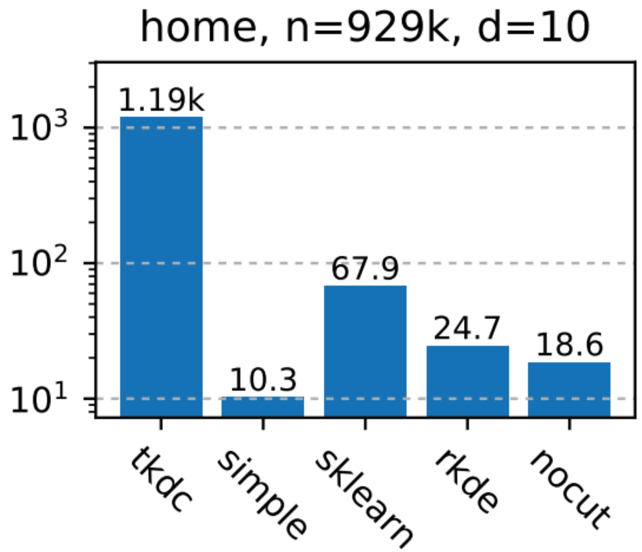
Classification: only need to tell whether above or below target

Don't need to compute exact densities!

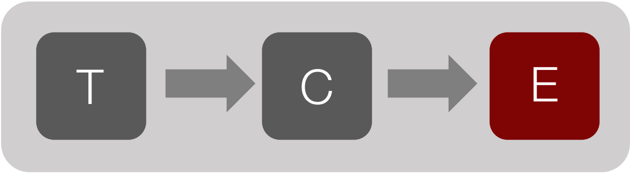
Use branch and bound: 2 orders of magnitude speedup

[Gan and Bailis, SIGMOD 2017]



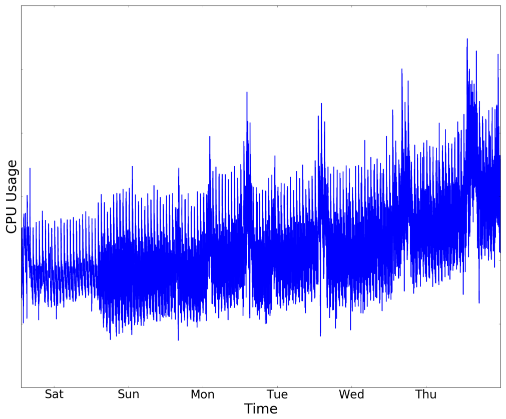


Efficient Parameter Search in Time Series Visualization



Time Series Smoothing

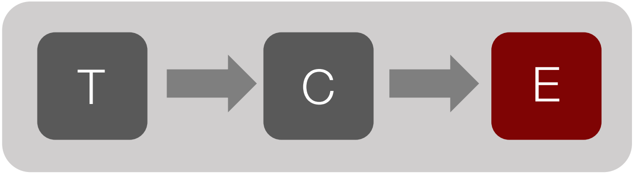
Raw time series are hard to read



Original: noisy

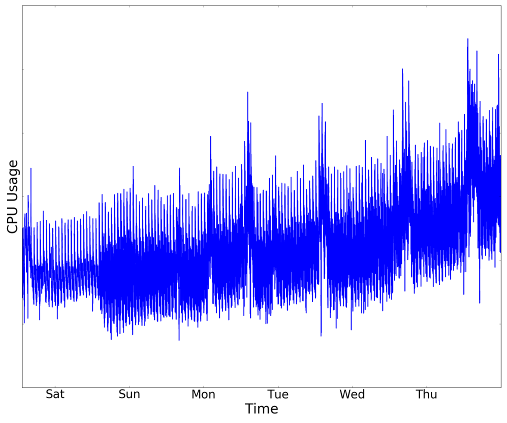
[Rong and Bailis, VLDB 2017]

Efficient Parameter Search in Time Series Visualization

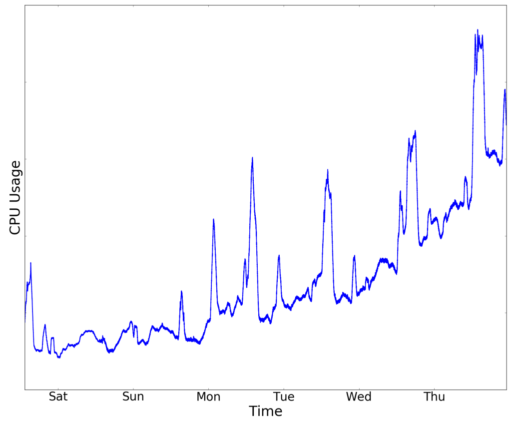


Time Series Smoothing

Raw time series are hard to read; Smoothing can help!



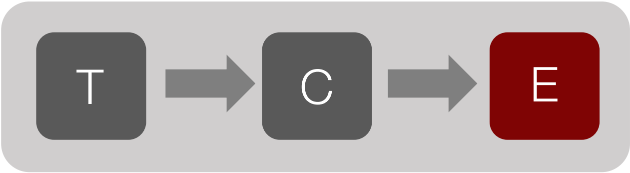
Original: noisy



Good: retains "outlyingness"

[Rong and Bailis, VLDB 2017]

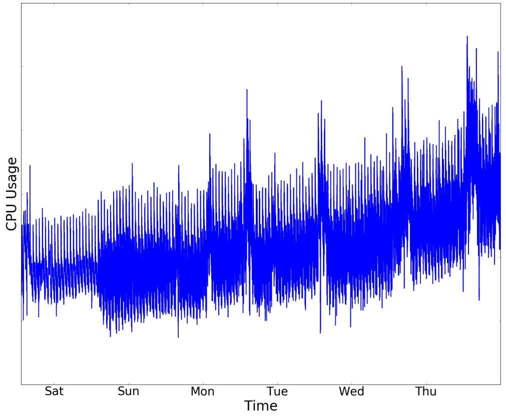
Efficient Parameter Search in Time Series Visualization



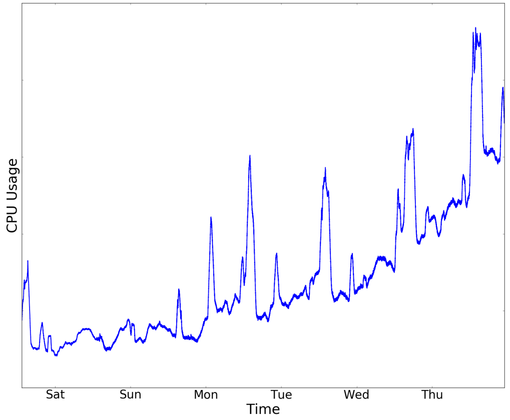
Time Series Smoothing

Raw time series are hard to read; Smoothing can help!

Challenge: Automatically choose smoothing parameters



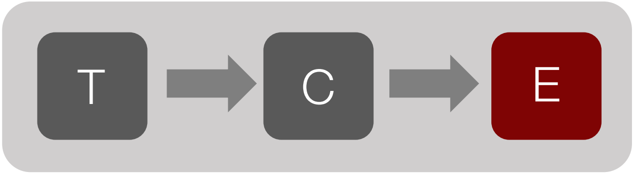
Original: noisy



Good: retains “outlyingness”

[Rong and Bailis, VLDB 2017]

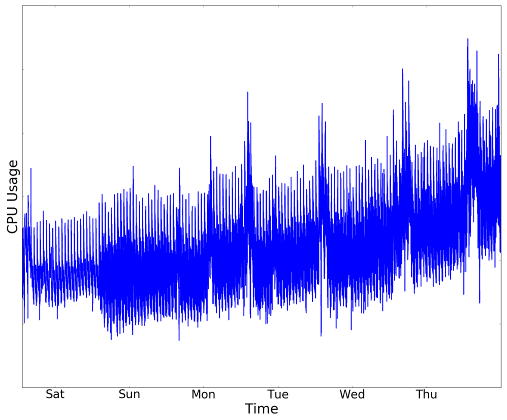
Efficient Parameter Search in Time Series Visualization



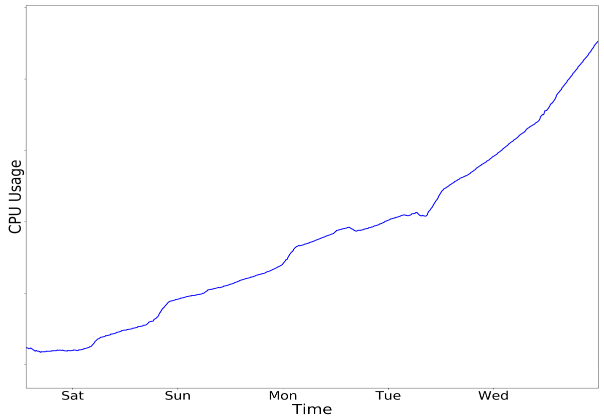
Time Series Smoothing

Raw time series are hard to read; Smoothing can help!

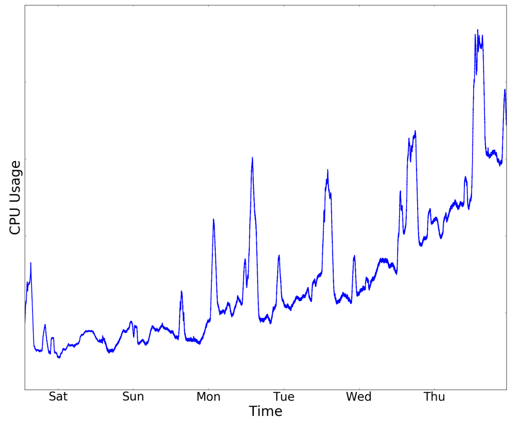
Challenge: Automatically choose smoothing parameters



Original: noisy



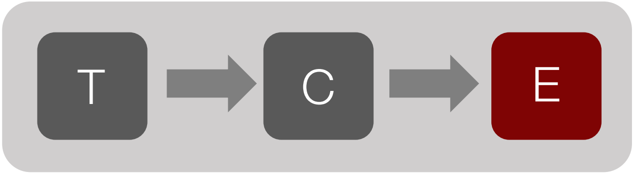
Bad: loses "outlyingness"



Good: retains "outlyingness"

[Rong and Bailis, VLDB 2017]

Efficient Parameter Search in Time Series Visualization

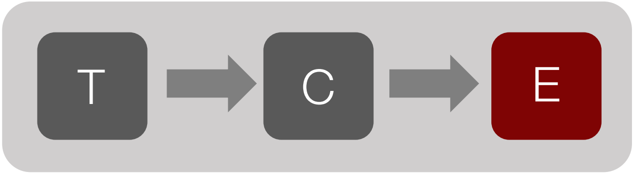


Time Series Smoothing

- Formulate as optimization problem: Smooth as much as possible while preserving long-term deviations

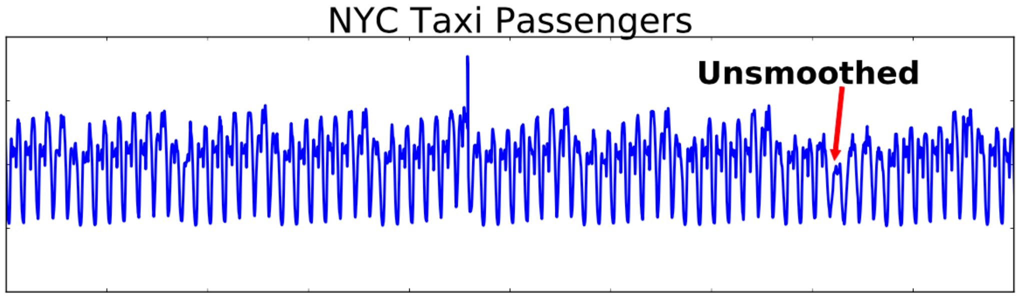
[Rong and Bailis, VLDB 2017]

Efficient Parameter Search in Time Series Visualization



Time Series Smoothing

- Formulate as optimization problem: Smooth as much as possible while preserving long-term deviations



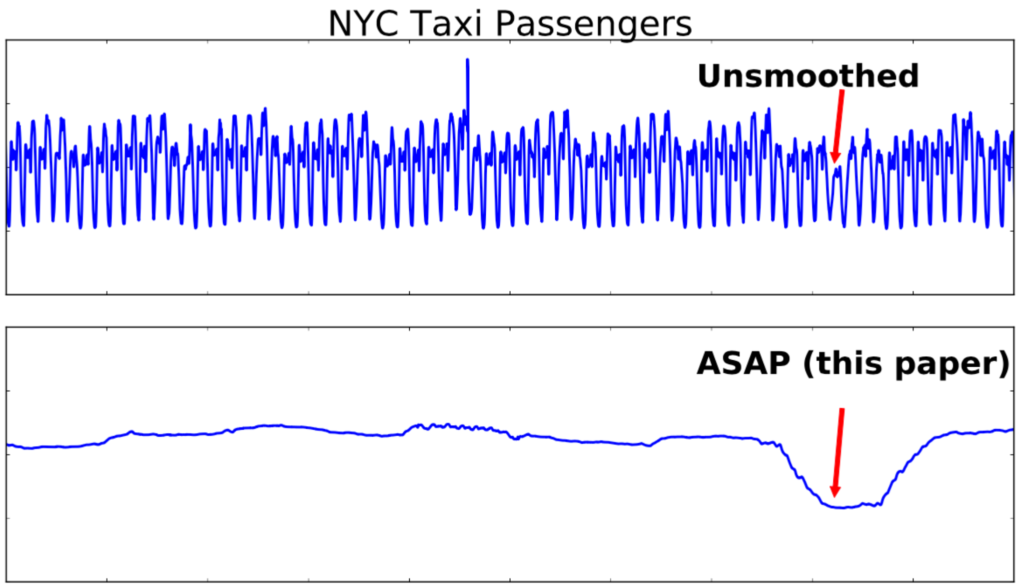
[Rong and Bailis, VLDB 2017]

Efficient Parameter Search in Time Series Visualization

T → C → E

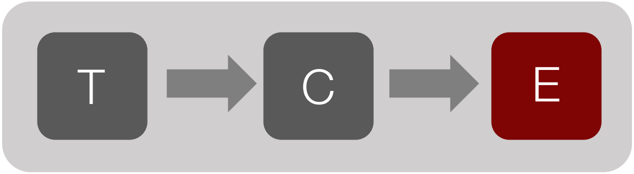
Time Series Smoothing

- Smooth as much as possible while preserving long-term deviations



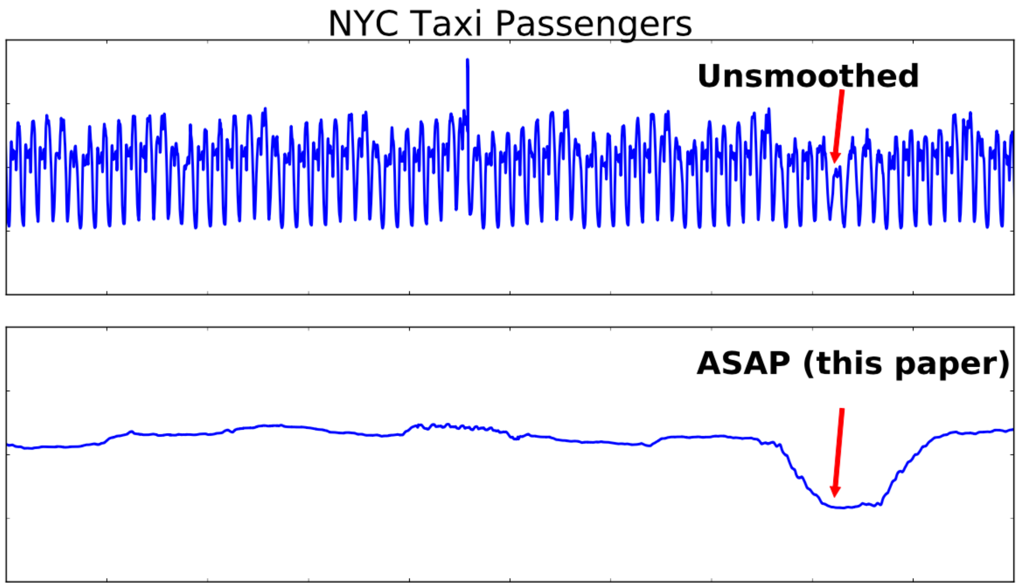
[Rong and Bailis, VLDB 2017]

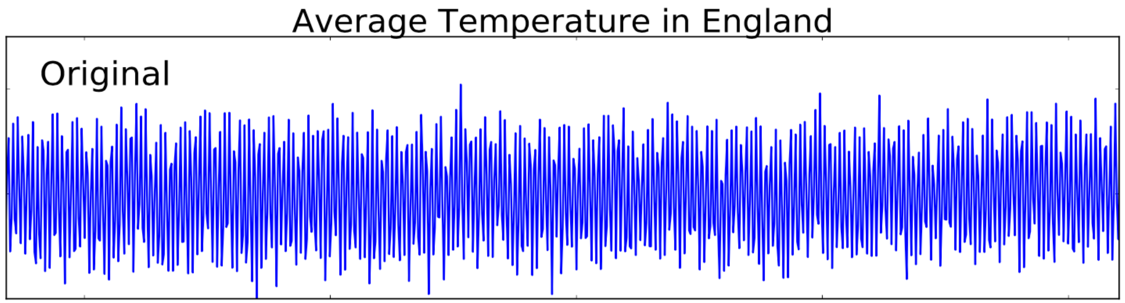
Efficient Parameter Search in Time Series Visualization



Time Series Smoothing

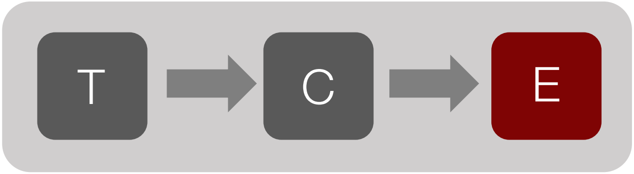
- Smooth as much as possible while preserving long-term deviations





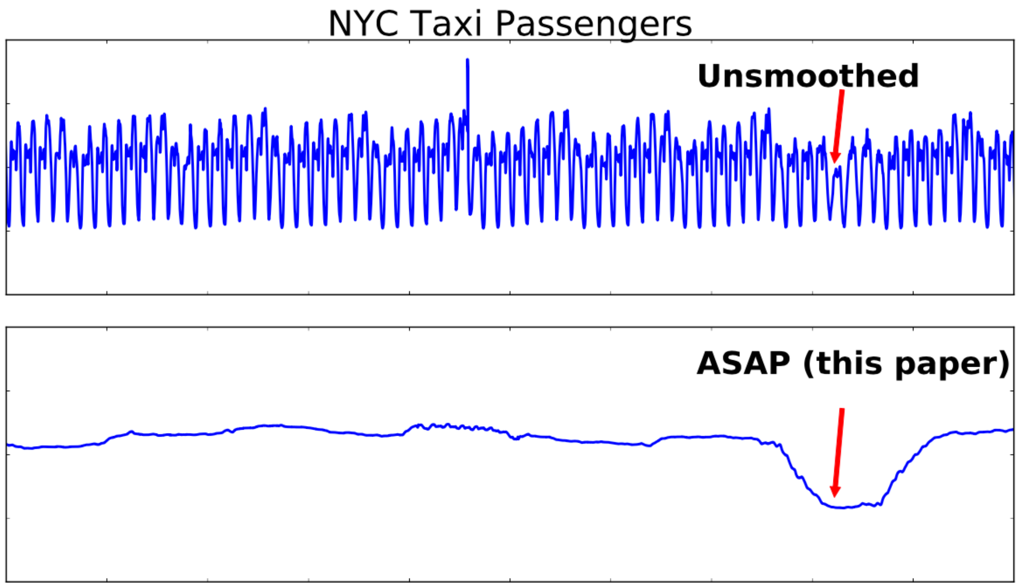
[Rong and Bailis, VLDB 2017]

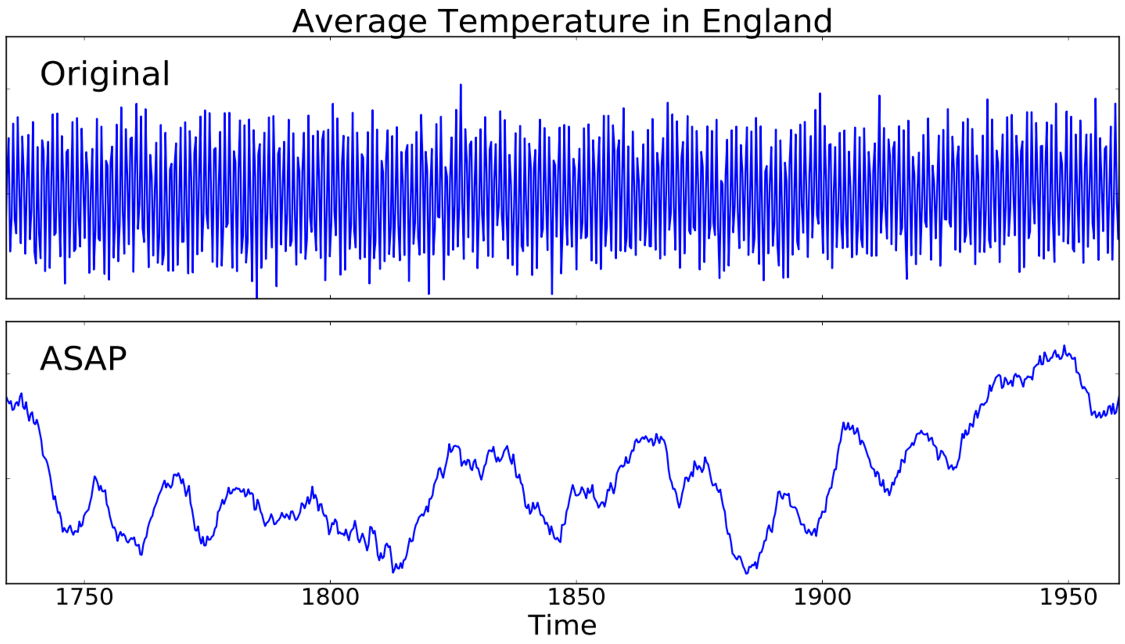
Efficient Parameter Search in Time Series Visualization



Time Series Smoothing

- Smooth as much as possible while preserving long-term deviations





[Rong and Bailis, VLDB 2017]

CIDR17, SIGMOD17: Overview of Preceding

Prioritizing Attention in Fast Data: Principles and Promise

Peter Bailis, Edward Gan, Kexin Rong, Sahaana Suri
Stanford InfoLab

ABSTRACT

While data volumes continue to rise, the capacity of human attention remains limited. As a result, users need analytics engines that can assist in prioritizing attention in this *fast data* that is too large for manual inspection. We present a set of design principles for the design of fast data analytics engines that leverage the relative scarcity of human attention and overabundance of data: return fewer results, prioritize iterative analysis, and filter fast to compute less. We report on our early experiences employing these principles in the design and deployment of MacroBase, an open source analysis engine for prioritizing attention in fast data. By combining streaming operators for feature transformation, classification, and data summarization, MacroBase provides users with interpretable explanations of key behaviors, acting as a search engine for fast data.

1. INTRODUCTION

In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention

MacroBase: Prioritizing Attention in Fast Data

Peter Bailis, Edward Gan, Samuel Madden[†], Deepak Narayanan, Kexin Rong, Sahaana Suri
Stanford InfoLab and [†]MIT CSAIL

ABSTRACT

As data volumes continue to rise, manual inspection is becoming increasingly untenable. In response, we present MacroBase, a data analytics engine that prioritizes end-user attention in high-volume *fast data* streams. MacroBase enables efficient, accurate, and modular analyses that highlight and aggregate important and unusual behavior, acting as a search engine for fast data. MacroBase is able to deliver order-of-magnitude speedups over alternatives by

However, the design and implementation of this infrastructure is challenging; current analytics deployments are a far cry from this potential. Today, application developers and analysts can employ a range of scalable dataflow processing engines to compute over fast data (over 20 in the Apache Software Foundation alone). However, these engines leave the actual implementation of scalable analysis operators that prioritize attention (e.g., highlighting, grouping, and contextualizing important behaviors within fast data) up to the application developer. This development is hard: fast data analyses must

Outline

Prioritizing Attention in Fast Data

Demo

Architecture + Usage

A Relational Algebra for MacroBase

Next Generation: Declarative Algebra

Is there a more general interface for composing MacroBase queries, and combining with external analytics operators?

Like SQL!

Our proposal: the `DIFF` operator
find the differences between two relations

MACRODIFF: MacroBase as SQL

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
```

Input

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
```

Transform

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
  SELECT * FROM
DIFF
  (SELECT * FROM md WHERE bd_percentile > 0.95) as outliers,
  (SELECT * FROM md WHERE bd_percentile <= 0.95) as inliers
```

Classify

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
  SELECT * FROM
DIFF
  (SELECT * FROM md WHERE bd_percentile > 0.95) as outliers,
  (SELECT * FROM md WHERE bd_percentile <= 0.95) as inliers
ON
  app_version, hw_make, hw_model, firmware_version
```

Attributes

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
  SELECT * FROM
DIFF
  (SELECT * FROM md WHERE bd_percentile > 0.95) as outliers,
  (SELECT * FROM md WHERE bd_percentile <= 0.95) as inliers
ON
  app_version, hw_make, hw_model, firmware_version
COMPARE BY risk_ratio(COUNT(*)) AS rr
```

Explain

MACRODIFF: MacroBase as SQL

```
# percentile defined by user
WITH SELECT *, percentile(battery_drain) as bd_percentile FROM mobile_data as md
  SELECT * FROM
DIFF
  (SELECT * FROM md WHERE bd_percentile > 0.95) as outliers,
  (SELECT * FROM md WHERE bd_percentile <= 0.95) as inliers
ON
  app_version, hw_make, hw_model, firmware_version
COMPARE BY risk_ratio(COUNT(*)) AS rr
WHERE rr > 3.0 and SUPPORT > 0.25
LIMIT 25;
```

MACRODIFF Demo

`DIFF`: composable with relational algebra

app_version	hw_make	hw_model	firmware_version	risk_ratio	support
v48	HTC	*null*	4.3.1	25x	20%
null	Lenovo	Lenovo_A390	5.1.1	10x	15%
v50	Emdoor	em_i8180	*null*	100x	7%

Return normalized relation

Schema of input relations retained, ratio and support attributes added

Composable with downstream SQL queries

`DIFF` is the new `CUBE`

`CUBE` lets you slice
and dice your data

`DIFF` tells you **how**
to slice and dice your data

Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals

Jim Gray — Microsoft — Gray@Microsoft.com
Adam Bosworth — Microsoft — AdamB@Microsoft.com
Andrew Layman — Microsoft — AndrewL@Microsoft.com
Hamid Pirahesh — IBM — Pirahesh@Almaden.IBM.com

Abstract: Data analysis applications typically aggregate data across many dimensions looking for unusual patterns. The SQL aggregate functions and the GROUP BY *operator produce zero-dimensional or one-dimensional answers. Applications need the N-dimensional generalization of these operators. This paper defines that operator, called the* data cube *or simply* cube. *The cube operator generalizes the histogram, cross-tabulation, roll-up, drill-down, and sub-total constructs found in most report writers. The*

points such as temperature, pressure, humidity, and wind velocity. Often these measured values are aggregates over time (the hour) or space (a measurement area).

Table 1: Weather

Time (UCT)	Latitude	Longitude	Altitude (m)	Temp (c)	Pres (mb)
27/11/94:1500	37:58:33N	122:45:28W	102	21	1009
27/11/94:1500	34:16:18N	27:05:55W	10	23	1024

Recent Work

Read our blog posts!
http://dawn.cs.stanford.edu/blog

- MacroBase motivation [CIDR 2017]
- MacroBase architecture, sketches [SIGMOD 2017]
- tKDC classification [SIGMOD 2017]
- NoScope video classification [VLDB 2017]
- ASAP time-series visualization [VLDB 2017]
- DROP dimensionality reduction [forthcoming]

Conclusion

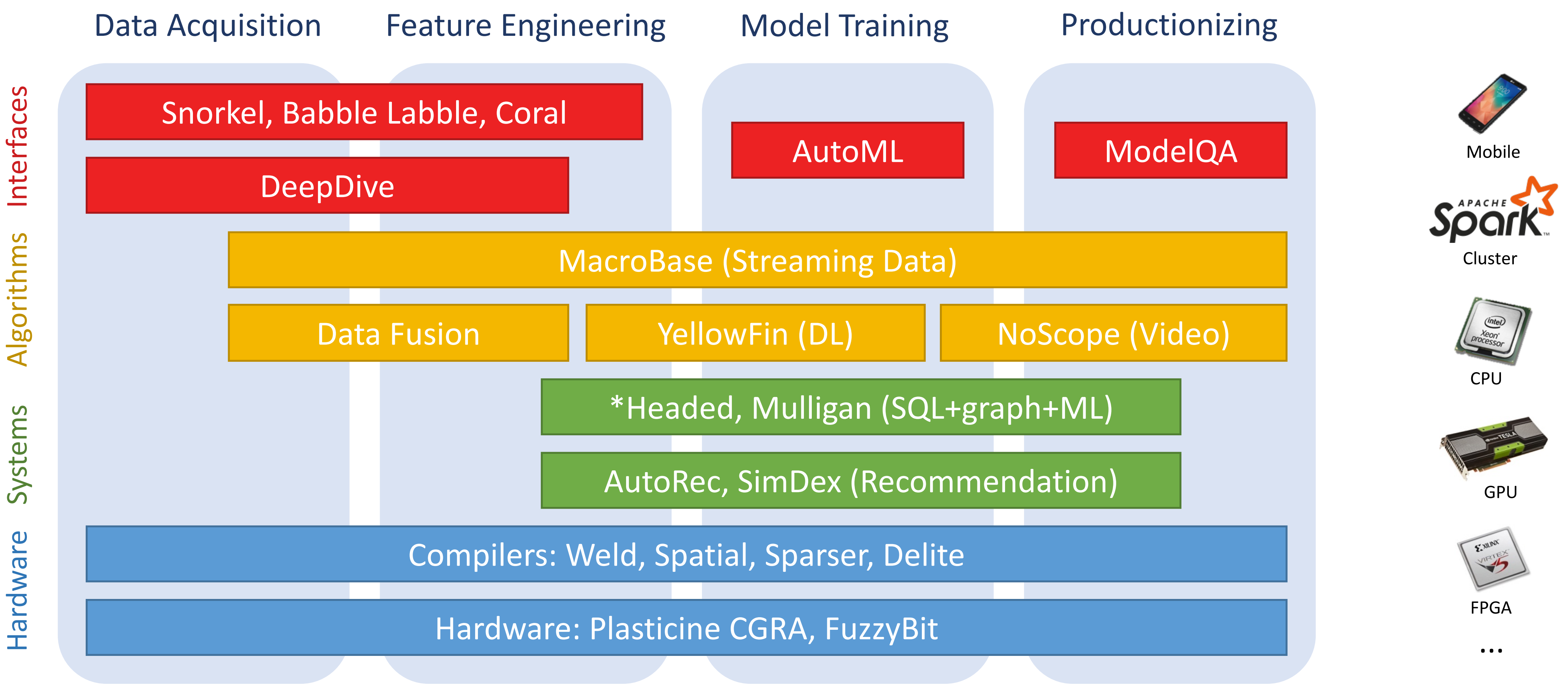
Increasing need for data monitoring demand new tools for prioritizing human attention; dataflow engines are not enough

MacroBase: combine feature extraction, classification, explanation in an end-to-end analytic monitoring engine

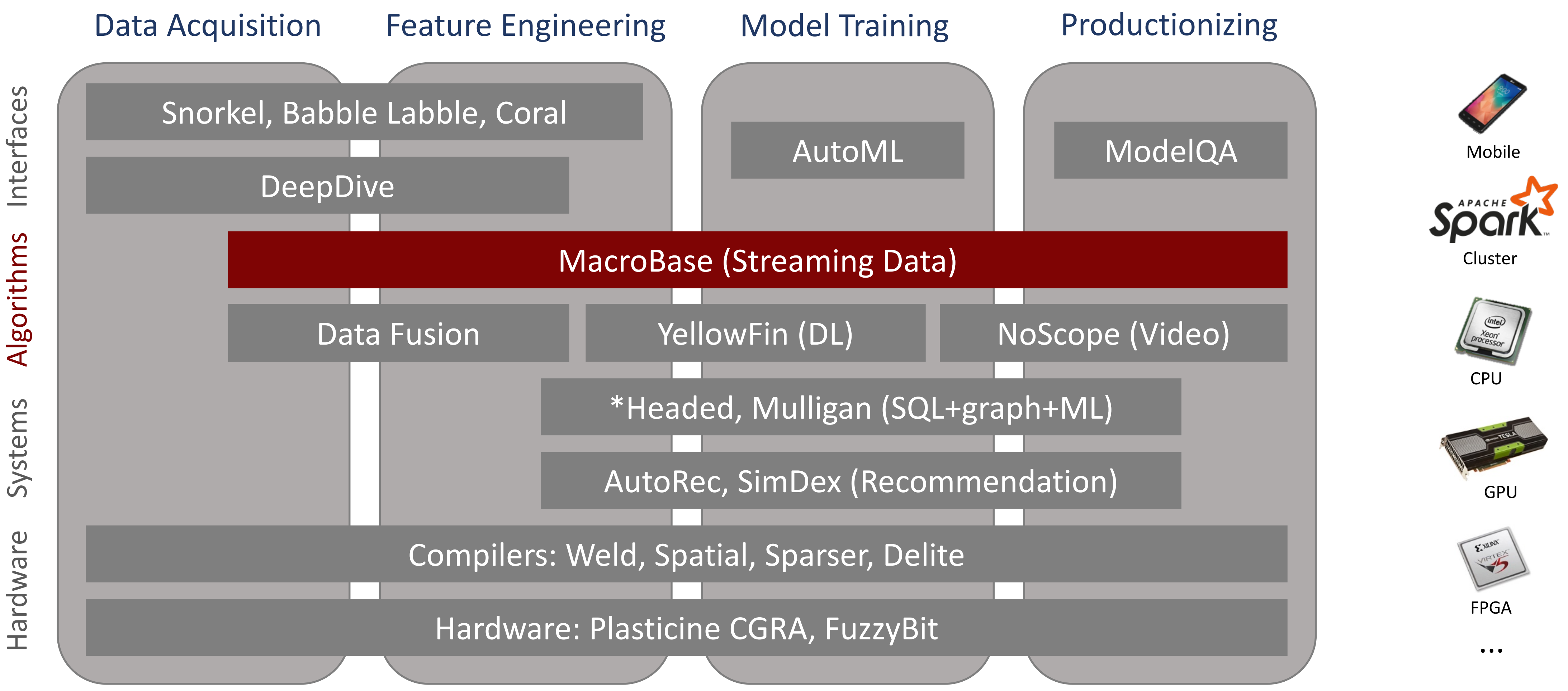
https://github.com/stanford-futuredata/macrobase

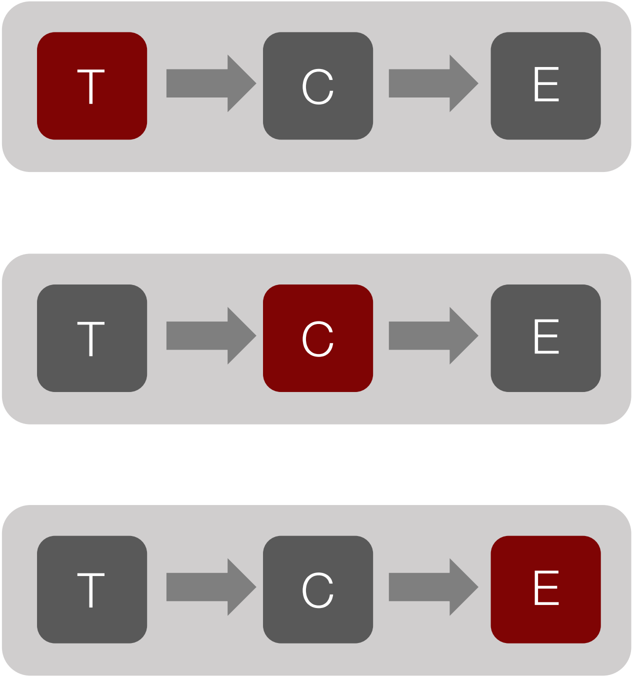
http://dawn.cs.stanford.edu/blog

DAWN Stack


Data Acquisition
Feature Engineering
Model Training
Productionizing
Interfaces
Algorithms
Systems
Hardware
Snorkel, Babble Labble, Coral
DeepDive
AutoML
ModelQA
MacroBase (Streaming Data)
Data Fusion
YellowFin (DL)
NoScope (Video)
*Headed, Mulligan (SQL+graph+ML)
AutoRec, SimDex (Recommendation)
Compilers: Weld, Spatial, Sparser, Delite
Hardware: Plasticine CGRA, FuzzyBit
Mobile
APACHE Spark
Cluster
CPU
GPU
FPGA
...

DAWN Stack


Data Acquisition
Feature Engineering
Model Training
Productionizing
Interfaces
Algorithms
Systems
Hardware
Snorkel, Babble Labble, Coral
DeepDive
AutoML
ModelQA
MacroBase (Streaming Data)
Data Fusion
YellowFin (DL)
NoScope (Video)
*Headed, Mulligan (SQL+graph+ML)
AutoRec, SimDex (Recommendation)
Compilers: Weld, Spatial, Sparser, Delite
Hardware: Plasticine CGRA, FuzzyBit
Mobile
Spark
Cluster
CPU
GPU
FPGA
…


T
C
E
T
C
E
T
C
E